MANUAL

AC500-S

Safety user manual V1.2.1

Original instructions

Table of contents

1 Introduction

1.1 Purpose

This safety user manual describes AC500-S safety PLC system. It provides detailed information on how to install, run, program and maintain the system correctly in functional safety applications up to SIL 3 according to IEC 61508:2010, SILCL 3 according to IEC 62061:2015 and performance level e (cat. 4) according to ISO 13849-1:2015.

ABB's AC500 series is a PLC-based modular automation solution that makes it easy to mix and match safety and non-safety I/O modules to meet automation market requirements.

1.2 Document history

Rev.	Description of version / changes	Who	Date
1.2.1	Various improvements in the text. Major changes: ● Chapters 3.4.7 and 3.5.7: New circuit examples for DX581-S and AI581-S were added. ● Chapter 4.1: Information about new Safety Engineering was added. ● Chapter 6.2: New check list item no. 23 for endianness checks was added.	ABB	24.03.2021
1.2.0	Various typos were corrected and various improvements in the texts and illustrations were made. Layout was changed to current ABB branding. Major changes: ● Chapter 4.3.7.1: New safety verification tool SVT was added. ● Safety modules are supported by AC500 V3 non-safety CPUs. Specific information on handling safety modules with non-safety CPUs transferred to appendices B + C. Appendix B contains all specific information about safety modules with V2 non-safety CPUs PM5xx. Appendix C contains all specific information about safety modules with V3 non-safety CPUs PM56xx. ● Chapter 3.1.2.6: "Firmware, boot code and boot project update" was updated. ● Assembly instructions of safety I/O modules were updated.	ABB	19.06.2020
1.1.0	Various typos were corrected. Various improvements in the text. Major changes: ● Information about SM560-S-FD-1(-XC) and SM560-S-FD-4(-XC) safety CPUs was added. ● Ch. 4.6.7: New PROFIsafe F-Device library SafetyDeviceExt_LV100_PROFIsafe_AC500_V27.lib was added. ● Ch. 4.6.8: New Safety library SafetyExt2_LV100_AC500_V27.lib was added. ● Detailed information about relevant standards was added. ● Checklists for AC500-S commissioning in Chapter 6 were updated.	ABB	16.03.2018

Rev.	Description of version / changes	Who	Date
1.0.5	Various typos were corrected. Minor improvements in the text and removal of screen shots for older versions of Automation Builder. Major changes: ● New PROFIsafe F-Host library SAFETY-BASE_PROFIsafe_LV200_AC500_V22.lib is used in the document. ● FAQ (Frequently Asked Questions) list was added. ● Ch. 2.4: Detailed safety values for AC500-S modules were provided. ● Ch. 4.3.6: "DANGER!" note was added to explain PROFIsafe Device_Fault bit usage. ● Ch. 6.3: New checklist item 9 was added.	ABB	23.10.2017
1.0.4	Various typos were corrected. Minor improvements in the text. Major changes: Licensing information was updated: ● Ch. 4.1: Notice Block with reference to PS501-S license installation removed. ● Ch. 4.2: Figure 63 updated (Programming workflow, step 2) was enhanced for the license handling of Automation Builder version V2.0.2 (or higher). ● Ch. 4.3.2: "Licence activation" was extended with additional licensing information for usage of Automation Builder version V2.0.2 (or higher). Additional information according to the new F-Host library "SAFETY-BASE_PROFIsafe_AC500_V22_Ext.lib" was added: ● Ch 4.6.1: Table for library "SAFETY-BASE_PROFIsafe_AC500_V22_Ext.lib" was updated. ● Ch. 4.6.3: The chapter was updated and renamed acc. to the new library name "SAFETYBASE_PROFIsafe_AC500_V22_Ext.lib". ● Ch. 6.2: Checklist item 20 was updated according to the new library name "SAFETYBASE_PROFIsafe_AC500_V22_Ext.lib".	ABB	27.03.2017
1.0.3	Various typos were corrected. Additional abbreviations were included in the abbreviation list. The entire document was re-styled: ● The yellow background on notices and recommendations was replaced by a light-grey background because of document standardization. ● "DANGER" and "NOTICE" symbols were replaced by standard symbols from German Standard DIN 4844-2 in text boxes. The text was changed in the document: ● More standard terms are now used in the document. ● Values for storage and transport temperatures were extended. ● Vertical mounting option (with derating) is added for SM560-S Safety CPU and corrected for DI581-S and AI581-S Safety I/O modules. ● LREAL is not supported by SM560-S Safety CPUs and was removed from the document. ● POU SF_MAX_POWER_DIP_GET description was modified. ● "DANGER" text box was added for POU SF_DPRAM_PM5XX_S_SEND to explain limitations for POU usage. ● F_WD_Time2 and Device_WD2 term definitions in Chapter 5.3 were corrected. ● "F_Host_WD" was replaced with "the value set using SF_WDOG_TIME_SET" inside of "NOTICE" box in Chapter 5.3	ABB	28.05.2015
1.0.2	Words "Original Instructions" have been added to document title	ABB	17.04.2015

Rev.	Description of version / changes	Who	Date
1.0.1	Minor typos were corrected. TÜV SÜD certificate was added. The text was changed in the document: ● Safety I/O inputs and outputs are not electrically isolated from the other electronic circuitry of the module. ● The safety values for safety outputs of DX581-S (-XC) module are only valid if the parameter "Detection" is set to "On". ● DC (diagnostic coverage) for DX581-S (-XC) module shall be ≥ 94 %. ● The clarification was added that the boot project update on SM560-S is possible only if no boot project is loaded on SM560-S. ● Not more than one communication error (CE_CRC or Host_CE_CRC output signals become equal to TRUE) per 100 hours is allowed to be acknowledged by the operator using OA_C input signal without consulting the responsible safety personnel. ● SM560-S cycle time shall be included three times instead of two times in Safety Function Response Time calculation. ● The values for input delay accuracy in Safety Function Response Time calculation were updated. ● Update of Appendix A with system data for AC500-S-XC.	ABB	08.03.2013
1.0.0	First release	ABB	19.12.2012

1.3 Validity

The data and illustrations found in this documentation are not binding. ABB reserves the right to modify its products in line with its policy of continuous product development.

1.4 Important user information

This documentation is intended for qualified personnel familiar with functional safety. You must read and understand the safety concepts and requirements presented in this safety user manual prior to operating AC500-S safety PLC system.

The following special notices may appear throughout this documentation to warn of potential hazards or to call attention to specific information.

> **DANGER!**
> The notices referring to your personal safety are highlighted in the manual by this safety alert symbol, which indicates that death or severe personal injury may result if proper precautions are not taken.

> **NOTICE!**
> This symbol of importance identifies information that is critical for successful application and understanding of the product. It indicates that an unintended result can occur if the corresponding information is not taken into account.

1.5 Definitions, expressions, abbreviations

1oo2	One-out-of-Two safety architecture, which means that it includes two channels connected in parallel, such that either channel can process the safety function.
AC500	ABB non-safety PLC
AC500-XC	ABB non-safety PLC suitable for extreme environmental conditions
AC500-S	ABB safety PLC for applications up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1)
AC500-S-XC	ABB safety PLC for applications up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1) suitable for extreme environmental conditions
ADC	Analog to digital converter
AOPD	Active opotoelectronic protective device
Automation Builder	Integrated engineering suite for ABB PLCs, including IEC 61131-3 editor CODESYS
CCF	Common cause failure
Control Builder Plus PS501	Integrated engineering suite for ABB PLCs, including IEC 61131-3 editor CODESYS, predecessor of Automation Builder
CPU	Central processing unit
CRC	Cyclic redundancy check. A number derived from and stored or transmitted with a block of data in order to detect data corruption.
DC	Diagnostic coverage
DPRAM	Dual-ported random access memory
DUT	Data unit type
IEC	International electro-technical commission standard
EDM	External device monitoring signal, which reflects the state transition of an actuator
EMC	Electromagnetic compatibility
EN	European norm (european standard)
EPROM	Erasable programmable read-only memory
ESD	Electro static discharge
ESPE	Electro-sensitive protective equipment (for example a light curtain)
F-Host	Data processing unit that is able to perform a special protocol and to service the "black channel" ➮ *[3]*
F-Device	Passive communication peer that is able to perform the special protocol, usually triggered by the F-Host for data exchange ➮ *[3]*
F-Parameter	Fail-safe parameter as defined in ➮ *[3]*
FAQ	Frequently asked questions
FB	Function block
FBD	Function block diagram (IEC 61131 programming language)
Flash memory	Non-volatile computer storage chip that can be electrically erased and reprogrammed
FV	Fail-safe value
GSDML	Generic station description markup language
ID	Identification
IO controller	Controller that controls the automation task in PROFINET context

IO device	Field device, monitored and controlled by an IO controller in PROFINET context
iParameter	Individual safety device parameter
LAD	Ladder logic diagram (IEC 61131 programming language)
Loop-back	The programmable routing feature of a bus device re-routes unintentionally an F-Host message back to the F-Host, which expects a message of the same length (refer to *www.profisafe.net* for further details).
LSB	Least significant bit
MSB	Most significant bit
MTBF	Mean time between failures
MTTF	Mean time to failure
Muting	Muting is the intended suppression of the safety function. This is required, e.g. when transporting the material into the danger zone.
NC	Break contact. Normally-closed contacts disconnect the circuit when the relay is activated; the circuit is connected when the relay is inactive.
NO	Make contact. Normally-open contacts connect the circuit when the relay is activated; the circuit is disconnected when the relay is inactive.
OEM	Original equipment manufacturer
OSSD	Output signal switching device
Passivation	The passivation is the special state of safety I/O modules which leads to the delivery of safe substitute values, which are ‘0’ values in AC500-S, to the safety CPU.
PC	Personal computer
PELV	Protective extra low voltage
PES	Programmable electronic system (refer to IEC 61508)
PFD	Probability of failure on demand
PFH	Probability of failure per hour
PL	Performance level according to ISO 13849-1
PLC	Programmable logic controller
POU	Program organization unit
Power cycle	Power cycle means to power off the safety CPU, wait for at least 1.5 s and power on the safety CPU again.
PROFIsafe	Safety-related bus profile of PROFIBUS DP/PA and PROFINET IO for communication between the safety program and the safety I/O in the safety system
PROFINET	Industrial technical standard for data communication over Industrial Ethernet
Proof Test Interval	The proof test is a periodic test performed to detect failures in a safety-related system so that, if necessary, the system can be restored as close as possible to its previous new state. The time period between these tests is the proof test interval.
PS	Programming system
PTC	Positive temperature coefficient
RAM	Random access memory
Reintegration	It is the process of switching from substitute values "0" to the process data.
Safety variable	It is a variable used to implement a safety function in a safety-related system
SCA	Safety code analysis - ABB software tool to automatically check CODESYS Safety programming rules

SD card	Secure digital memory card
SELV	Safety extra low voltage
SFRT	Safety function response time
SIL	Safety integrity level (IEC 61508)
SILCL	SIL claim limit (IEC 62061)
ST	Structured text (IEC 61131 programming language)
SVT	Safety Verification Tool - ABB software tool to verify the AC500-S safety configuration in Automation Builder
TÜV	Technischer Überwachungs-Verein (technical inspection association)
TWCDT	Total worst case delay time
ULP	Unit in the last place, which is the spacing between floating-point numbers, i.e., the value the least significant bit represents if it is 1 (refer to *http://en.wikipedia.org/wiki/Unit_in_the_last_place* for more details).
WLAN	Wireless local area network

1.6 Functional safety certification

The AC500-S safety modules are safety-related up to SIL 3 according to IEC 61508:2010, SILCL 3 according to IEC 62061:2015 and performance level e according to ISO 13849-1:2015, as certified by TÜV SÜD Rail GmbH (Germany).

The AC500-S is a safety PLC which operation reliability is significantly improved compared to a non-safety PLC using 1oo2 redundancy in the hardware and additional diagnostic functions in its hardware and software. The embedded safety integrity diagnostic functions are based on the safety standards current at the time of certification ⇘ *TÜV SÜD Rail Certification Report for AC500-S [2]*. These safety integrity tests include test routines, which are run during the whole operating phase, making the AC500-S safety PLC suitable for the safety machinery and process applications up to SIL 3 according to IEC 61508:2010, SILCL 3 according to IEC 62061:2015 and performance level e according to ISO 13849-1:2015.

> **!** **NOTICE!**
> Please refer to TÜV SÜD Rail Certification Report for AC500-S ⇘ *[2]* for a complete list of standards and further details, like versions of standards, etc.

The proof test interval for the AC500-S safety PLC is set to 20 years.

PFH, PFD, MTTFd, category and DC values from IEC 61508:2010, IEC 62061:2015 and ISO 13849-1:2015 for AC500-S safety modules satisfy SIL 3, SILCL 3 and PL e requirements ⇘ *Chapter 2.4 “Safety values” on page 18*.

1.7 References / related documents

[1] - Creation of safety-oriented applications with CODESYS V2.3 - Document version 1.8

[2] - TÜV SÜD Rail Certification Report for AC500-S Safety PLC, Version - 2018 (or newer), available at *www.abb.com/plc*

[3] - PROFIsafe - Profile for Safety Technology on PROFIBUS DP and PROFINET IO Profile part, related to IEC 61784-3-3, Version 2.4, March, 2007 (or newer)

[4] - AC500 user documentation for Automation Builder / Control Builder Plus, available at *www.abb.com/plc*

[5] - IEC 61131, 2003 (or newer), Programmable Controllers, Part 3 - Programming Languages

[6] - Computer Science and Engineering at University of California, Riverside, Chapter 14, Ch14_Floating Point Calculations and its drawbacks.pdf

[7] - User Examples with PLCopen Safety Functions, Version 1.0.1, 2008 (or newer)

[8] - PROFIsafe System Description, Version - Nov. 2007 (or newer)

[9] - PLCopen Safety: Concepts and Function Blocks, Version 1.0, 2006 (or newer)

[10] - ISO 13849-1: Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design, 2015 (or newer)

[11] - PROFIBUS Guideline: PROFIsafe - Environmental Requirements, V2.5, March 2007 (or newer)

[12] - PROFIBUS Guideline: Communication Function Blocks on PROFIBUS DP and PROFINET IO, V2.0, November 2005. Order No. 2.182 (or newer)

1.8 Applicable standards

Standard	Date	Title
IEC 61508	2010	Functional safety of electrical/electronic/programmable electronic safety-related systems
IEC 62061	2015	Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems
ISO 13849-1	2015	Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design
IEC 60204-1	2016	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
IEC 61496-1	2012	Safety of machinery - Electro-sensitive protective equipment
IEC 61511-1	2016	Functional safety - Safety instrumented systems for the process industry sector - Part 1: Framework, definitions, system, hardware and software requirements
IEC 61326-3-1	2017	EMC for functional safety
IEC 61131-2	2007	Programmable controllers - Part 2: Equipment requirements and tests
ISA-71.04-2013 Harsh group A	2016	Environmental Conditions for Process Measurement and Control Systems - Airborne Contaminants
IEC 60721-3-3	2002	Classification of environmental conditions - Part 3-3: Classification of groups of environmental parameters and their severities - Stationary use at weather protected locations
CISPR 16-1-2	2014	Specification for radio disturbance and immunity measuring apparatus and methods - Part 1-2: Radio disturbance and immunity measuring apparatus - Ancillary equipment - Conducted disturbances

Standard	Date	Title
CISPR 16-2-1	2017	Specification for radio disturbance and immunity measuring apparatus and methods - Part 2-1: Methods of measurement of disturbances and immunity - Conducted disturbance measurements
CISPR 16-2-3	2016	Specification for radio disturbance and immunity measuring apparatus and methods - Part 2-3: Methods of measurement of disturbances and immunity - Radiated
IEC 61000-4-2	2008	Electromagnetic compatibility (EMC) - Part 4-2: Testing and measurement techniques - Electrostatic discharge immunity test
IEC 61000-4-3	2010	Electromagnetic compatibility (EMC) - Part 4-3: Testing and measurement techniques - Radiated, radio-frequency, electromagnetic field immunity test
IEC 61000-4-4	2012	Electromagnetic compatibility (EMC) - Part 4-4: Testing and measurement techniques - Electrical fast transient/burst immunity test
IEC 61000-4-5	2017	Electromagnetic compatibility (EMC) - Part 4-5: Testing and measurement techniques - Surge immunity test
IEC 61000-4-6	2013	Electromagnetic compatibility (EMC) - Part 4-6: Testing and measurement techniques - Immunity to conducted disturbances, induced by radio-frequency fields
IEC 61000-4-8	2009	Electromagnetic compatibility (EMC) - Part 4-8: Testing and measurement techniques - Power frequency magnetic field immunity test
EN 55011	2016	Industrial, scientific and medical equipment - Radio-frequency disturbance characteristics - Limits and methods of measurement (CISPR 11:2015, modified)
IEC 60715	2017	Dimensions of low-voltage switchgear and controlgear - Standardized mounting on rails for mechanical support of switchgear, controlgear and accessories
IEC 60068-2-1	2009	Environmental testing - Part 2-1: Tests - Test A: Cold
IEC 60068-2-6	2007	Environmental testing - Part 2-6: Tests - Test Fc: Vibration (sinusoidal)
IEC 60068-2-27	2008	Environmental testing - Part 2-27: Tests - Test Ea and guidance: Shock
IEC 60068-2-30	2005	Environmental testing - Part 2-30: Tests - Test Db: Damp heat, cyclic (12 + 12 h cycle)
IEC 60068-2-52	2017	Environmental testing - Part 2-52: Tests - Test Kb: Salt mist, cyclic (sodium chloride solution)
IEC 60068-2-64	2008	Environmental testing - Part 2-64: Tests - Test Fh: Vibration, broadband random and guidance
IEC 60068-2-78	2012	Environmental testing - Part 2-78: Tests - Test Cab: Damp heat, steady state

NOTICE!
Contact ABB technical support for further details.

2 Overview of AC500-S safety PLC

2.1 Overview

The AC500-S is realized as 1oo2 system (both safety CPU and safety I/O modules) and can be used to handle safety functions with SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1) requirements in high-demand systems of safety machinery applications and low-demand systems of safety process applications. 1oo2 system includes two microprocessors. Each of them executes the safety logic in its own memory area and both compare the results of the execution. If a mismatch in the execution or an error is detected, the system goes to a safe state, which is described for each of the safety modules separately.

2.1.1 System

The AC500-S safety PLC is an integrated part of AC500 platform with a real common look & feel engineering approach. Due to a tight integration in AC500 PLC platform, the generic AC500 system characteristics (mechanics, programming, configuration etc.) are also valid for AC500-S safety modules.

All non-safety AC500 modules are considered to be interference-free modules for AC500-S safety PLC. In contrast to safety modules, interference-free modules are not used to perform safety functions. A fault in one of these modules does not influence the execution of the safety functions in a negative way.

The term "integrated safety" applied for AC500-S safety PLC and AC500 platform means:

- One PROFINET IO fieldbus is used for safety and non-safety communication.
- The same engineering environment with real look & feel is used for both safety and non-safety programming.
- The same hardware and wiring look & feel is used within safety and non-safety modules.
- The same diagnostics concept is used for safety and non-safety modules.

Fig. 1: Overview on ABB's AC500 family with safety and non-safety modules

1 **Non-safety communication module**
AC500 covers all common communications standards, such as Ethernet, EtherCAT, PROFINET IO, PROFIBUS DP, CANopen, DeviceNet, Modbus TCP, Modbus serial, Serial, ABB CS31 and PROFIsafe via PROFINET. Combinable to form optimally scaled network nodes, ABB's AC500 is suitable for both small-scale applications and large-scale industrial systems.

2 **Safety CPU**
Safety CPUs certified up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1). An array of features such as system diagnostics provided via LEDs and onboard display of non-safety CPUs provides the added diagnostic concept required for integrated safety.

3 **Non-safety CPU**
ABB's complete AC500 range of non-safety CPUs can be used with safety CPU to create customized solutions - even for the most challenging requirements. The programming of safety and non-safety applications is offered via a non-safety PLC interface.

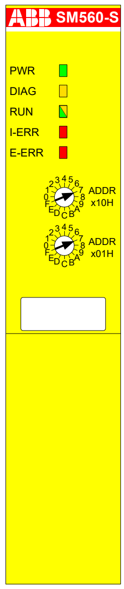
4 **Safety I/O module**
Safety I/O modules certified up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1). Features such as channel-wise error diagnostics and the flexibility to choose between channel-wise or module switch-off in case of channel error make working safely easier.

5 **Non-safety I/O module**
With ABB's non-safety I/O modules, the complete S500 and S500-eCo I/O module range can be connected to the non-safety PLC. A wealth of functions in AC500 configurable I/O modules allows getting the customized and low-priced solutions to optimize industrial applications.

2.1.2 Safety components

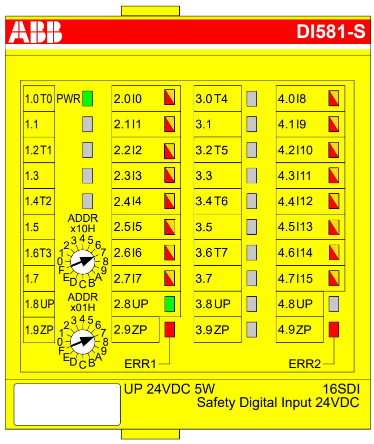
The AC500-S safety PLC includes the following safety-related hardware components.

SM560-S /
SM560-S-FD-1 /
SM560-S-FD-4



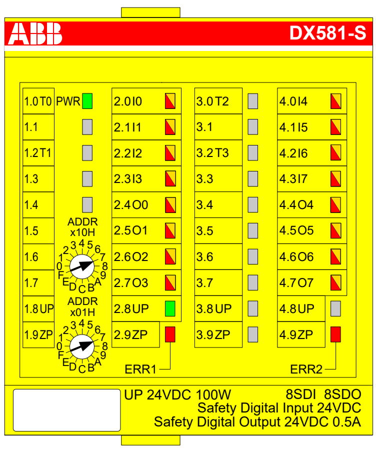
Safety CPU (safety module) for up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1) safety applications.

DI581-S



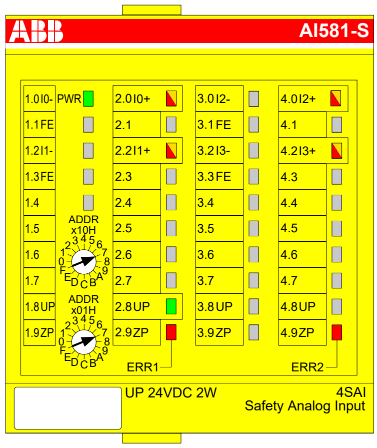
Safety binary input module DI581-S with 16 safety input channels (up to SIL 2 or PL d) or 8 safety input channels (up to SIL 3 or PL e) with 8 test pulse output channels.

DX581-S



Safety binary input/output module DX581-S with 8 safety output channels (up to SIL 3 or PL e) and 8 safety input channels (up to SIL 2 or PL d) or 4 safety input channels (up to SIL 3 or PL e) with 4 test pulse output channels.

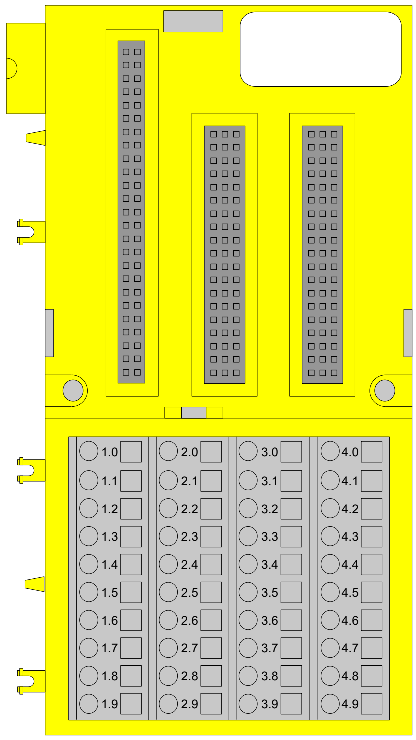
AI581-S



Safety analog input module AI581-S with 4 safety current input channels 0 ... 20 mA (up to SIL 2 or PL d) or 2 safety current input channels (up to SIL 3 or PL e).

The following interference-free component shall be used for mounting safety I/O modules:

TU582-S



Spring-type terminal unit TU582-S for safety I/O modules.

2.2 Intended use

The user shall coordinate usage of ABB AC500-S safety components in his applications with the competent authorities and get their approval. ABB assumes no liability or responsibility for any consequences arising from the improper use:

- Non-compliance with standards and guidelines
- Unauthorized changes to equipment, connections and settings
- Use of unauthorized or improper equipment
- Failure to observe the safety instructions in this guide

2.3 Safety loop

The safety loop, to which the AC500-S safety PLC belongs, consists of the following three parts: sensors, safety PLC and actuators.

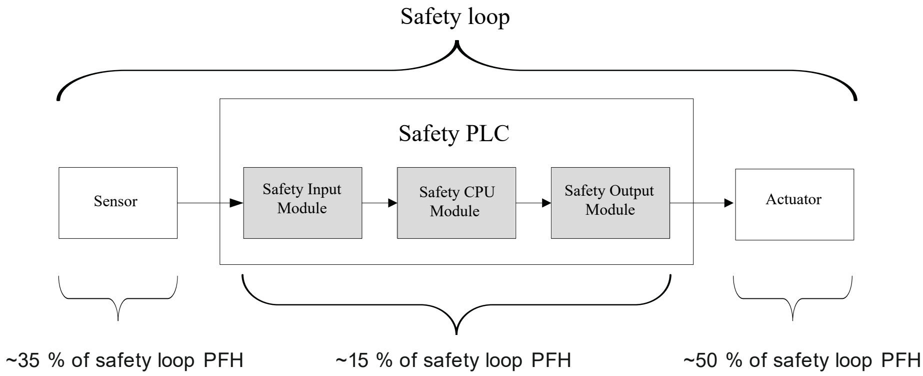


Fig. 2: Typical safety loop with AC500-S safety PLC

For the calculation of the PFH/PFD values of an exemplary safety system, a maximum of 15 % is assumed for the safety PLC.

2.4 Safety values

Table 1: The following safety values shall be used for AC500-S safety modules:

Type	**SIL(1)** **SILCL(2)**	**PL(3)**	**DC(4)**	**MTTFd(5)**	**PFHd(6)**	**PFHd(7)**	**PFDg(8)**	**T1(9)**	**SFF(10)**	**β(11)**
SM560-S(-XC) / SM560-S-FD-1(-XC) / SM560- S-FD-4(-XC)	3	e	97	1280	1.90E-09	8.95E-11	7.90E-06	20	98	2
AI581-S(-XC)	3	e	97	920	2.95E-09	4.50E-10	3.80E-05	20	99	2
DI581-S(-XC)	3	e	95	2270	1.45E-09	4.40E-10	3.70E-05	20	98	2
Inputs of DX581-S(-XC)	3	e	94	2250	1.45E-09	4.50E-10	3.80E-05	20	98	2
Outputs of DX581-S(-XC) with parameter Detection = "On"	3	e	94	1985	1.60E-09	4.50E-10	3.80E-05	20	99	2
Outputs of DX581-S(-XC) with parameter Detection = "Off"	2	d	85	200	1.19E-08	1.08E-08	4.70E-04	20	on request	2

(1) - SIL (safety integrity level) according to IEC 61508

(2) - SILCL (safety integrity level claim limit) according to IEC 62061

(3) - PL (performance level) according to ISO 13849-1

(4) - Diagnostic coverage, % (refer to ISO 13849-1)

(5) - Mean time to failures (years) dangerous according to ISO 13849-1

(6) - Probability of dangerous failure per hour according to IEC 62061

(7) - Probability of dangerous failure per hour according to IEC 61508 (High demand mode)

(8) - Average probability of failure to perform its design function on demand according to IEC 61508 (Low demand mode)

(9) - Proof test interval - mission time - lifetime years

(10) - SFF (safe failure fraction), % according to IEC 61508

(11) - β (beta factor), % for common cause failures according to IEC 61508

DANGER!
The average temperature for both the extended temperature range (-40 ... +70 °C) as well as for normal temperature range (0 ... +60 °C) is defined to +40 °C.

Ensure that average operating temperature for used AC500-S modules does not exceed +40 °C.

2.5 Qualified personnel

AC500-S safety PLC may only be set up and used in conjunction with this documentation.

Safety application engineer of AC500-S safety PLC
Commissioning and operation of AC500-S safety PLC may only be performed by the qualified personnel who is authorized to commission safety devices, systems and circuits in accordance with established functional safety practices and standards.

The following basic knowledge of AC500 system is required to correctly understand this AC500-S safety user manual:

- AC500 automation system.
- Automation Builder / Control Builder Plus programming environment (system configuration and programming in ST, LAD and FBD programming languages).

2.6 Lifecycle

All AC500-S safety modules have a maximum life of 20 years. This means that all AC500-S safety modules shall be taken out of service or replaced by new AC500-S safety modules at least one week before the expiry of 20 years (counted from the date of delivery by ABB).

2.7 Installation of safety modules

The following rules shall be taken into account for installing safety modules:

- The installation must be done according to the documentation with appropriate facilities and tools.
- The installation of the devices may be done only in de-energized condition and carried out by the qualified personnel.

- The general safety regulations and applicable national safety regulations shall be strictly observed.
- The electrical installation shall be carried out in accordance with relevant regulations.
- Take the necessary protective measures against static discharge.

2.8 Exchange of modules

SM560-S / SM560-S-FD-1 / SM560-S-FD-4 safety CPU automatically detects an exchange of safety I/O modules during the system start-up. The overall system (safety CPU and PROFIsafe features of unique addresses for safety devices ↯ *[3]*) provides a mechanism to automatically ensure that exchanged safety modules are operated with correct parameters and incompatible module types are rejected. No unsafe state is possible if wrong safety I/O module type is put on the given terminal unit TU582-S.

2.9 AC500-S restart behavior

When SM560-S / SM560-S-FD-1 / SM560-S-FD-4 safety CPU is restarted by a power cycle, the previously saved error information is lost. The safety I/O modules receive their parameter sets each time during system start-up. The safety CPU is able to reintegrate safety I/O modules using PROFIsafe start-up behavior ↯ *[3]*. If your process does not allow an automatic start-up after power cycle, you must program a restart protection in the safety program. The safety process data outputs must be blocked until manually acknowledged. These safety outputs must not be enabled until it is safe to do so and faults were corrected.

2.10 Replacing AC500-S safety PLC components

When replacing software components on your programming device or PC with a newer version, you must observe the notes regarding upward and downward compatibility in the documentation and readme files for these products.

Hardware components for AC500-S (safety CPU and safety I/Os) are replaced in the same way as in a non-safety AC500 automation system.

2.11 Environmentally friendly disposal

All AC500-S safety components from ABB are designed with a minimal environment pollution effect. To enable environmentally friendly disposal of AC500-S safety components, they can be partially disassembled to separate various components from each other. Disposal of those materials shall be done in accordance with applicable national and international laws.

2.12 Safe communication

Safety data are transferred between safety CPU and safety I/Os using PROFIsafe profile ↯ *[3]*. SM560-S / SM560-S-FD-1 / SM560-S-FD-4 safety CPU needs a non-safety CPU to communicate to safety I/O modules. All safety-related communication takes place through the non-safety CPU using a "black channel" principle of data transmission ↯ *[3]*.

The communication of safety CPU to remote safety I/O modules is done using PROFINET IO field bus with a PROFIsafe profile for safe data transmission ↯ *[3]*. Safety and non-safety I/O modules can be mixed on a local I/O bus both in central and remote configuration. PROFINET IO controller communication module (CM579-PNIO) shall be used on non-safety CPUs as a part of the "black channel" to transfer safety data to PROFINET IO devices. PROFINET devices CI501, CI502, CI504 and CI506 can be used to attach safety I/O modules in remote configurations.

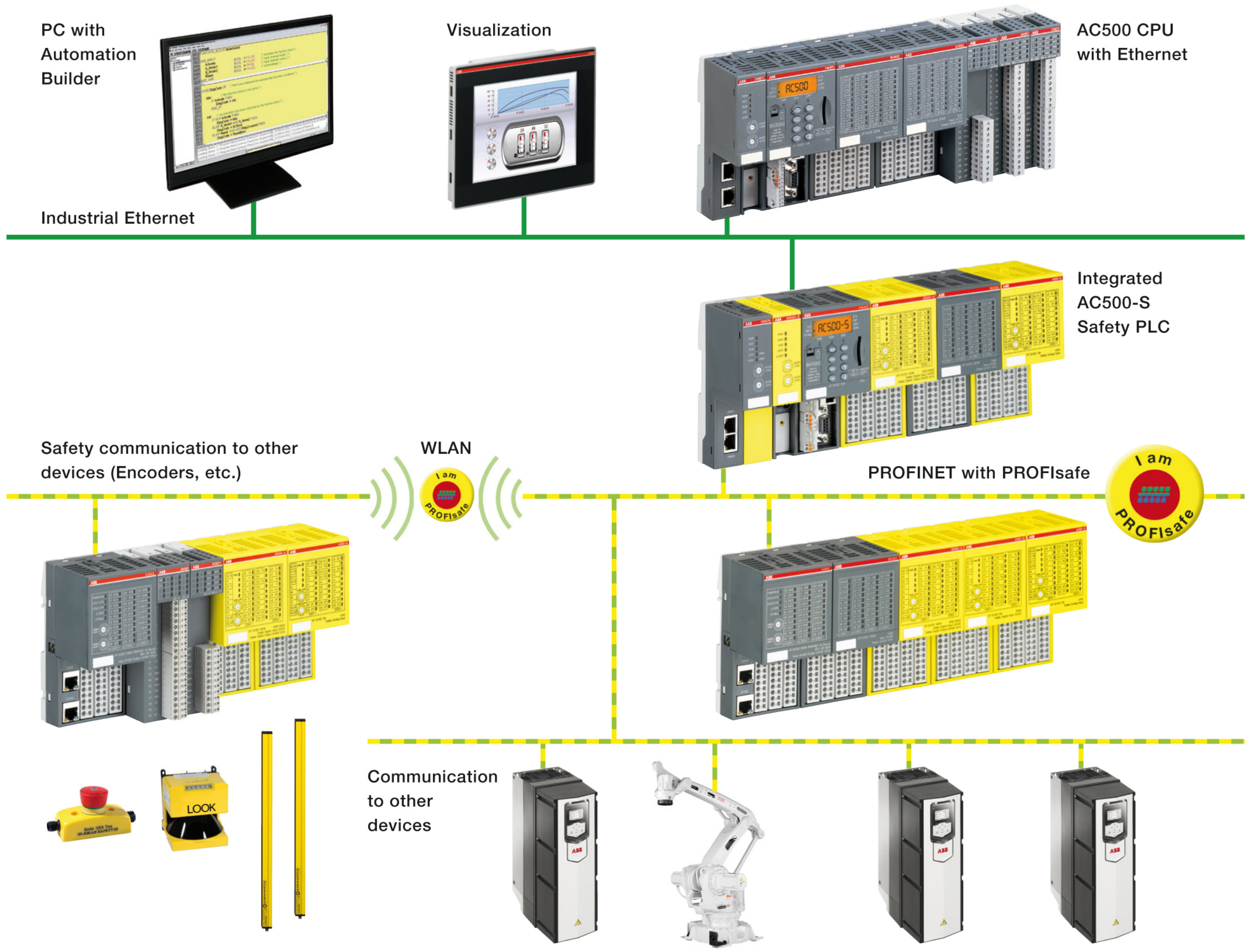


Fig. 3: Possible AC500-S system setup with PROFINET/PROFIsafe for remote safety I/Os, sensors and actuators

PROFINET/PROFIsafe communication between AC500-S safety CPUs is supported using CM589-PNIO(-XC) and/or CM589-PNIO-4(-XC) PROFINET IO device communication modules together with SM560-S-FD-1(-XC) and/or SM560-S-FD-4(-XC) safety CPUs with F-Device functionality on one side and CM579-PNIO(-XC) with any AC500-S safety CPU on the other side (Fig. 4 on page 22). SM560-S-FD-1(-XC) and SM560-S-FD-4(-XC) safety CPUs are able to exchange up to 384 bytes of safety data with F-Hosts (3rd party PROFIsafe F-Hosts are supported as well) using PROFINET/PROFIsafe. SM560-S-FD-1(-XC) with F-Device(s) supports safe communication to maximum one F-Host. SM560-S-FD-4(-XC) with F-Device(s) supports safe communication to maximum four F- Hosts. Fig. 4 shows that using SM560-S-FD-1 and SM560-S-FD-4 safety CPUs with additional F-Device functionality one can establish safe CPU to CPU communication between different control stations on PROFINET/PROFIsafe. SM560-S-FD-4 safety CPUs can simultaneously communicate not only with 1 IO controller/F-Host (Master) but with up to 4 IO controllers/F-Hosts (Masters). In addition to SM560-S-FD-1 and SM560-S-FD-4 safety CPUs, CM589-PNIO and CM589-PNIO-4 PROFINET IO device communication modules are needed to establish PROFINET connectivity as "black channel", respectively, to 1 or up to 4 PROFINET IO controllers.

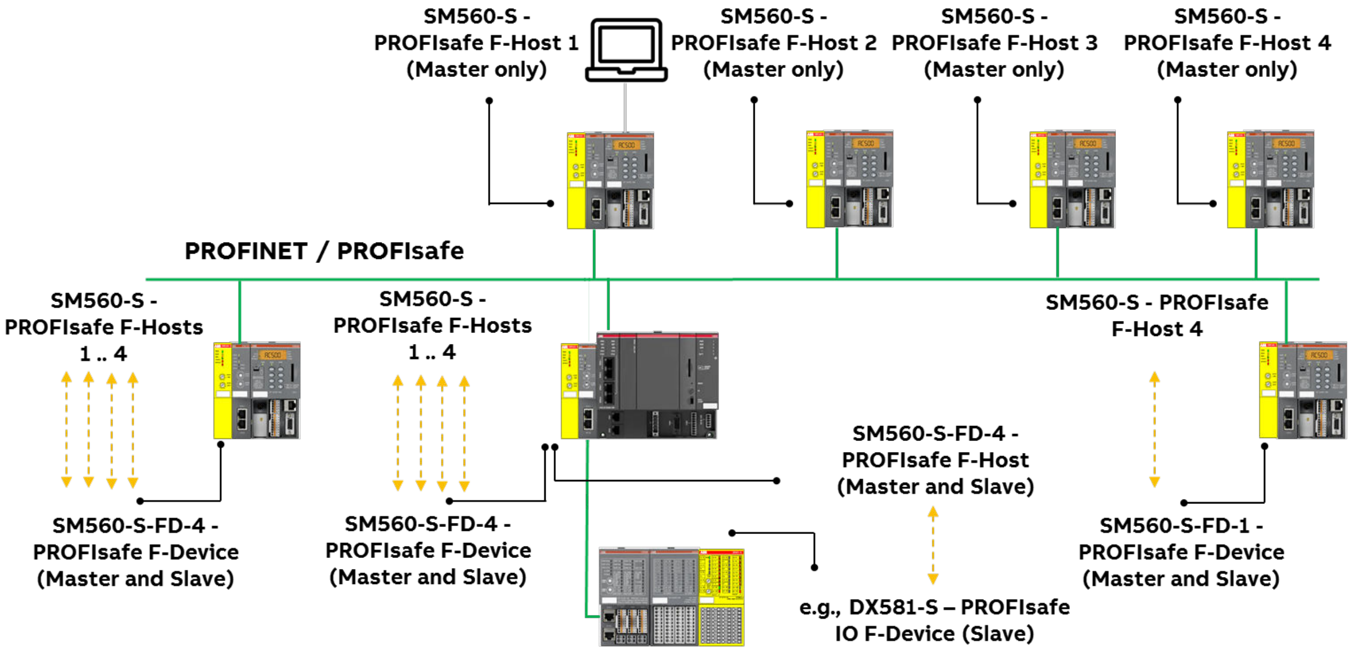


Fig. 4: Exemplary setup for safe CPU to CPU communication between various safety CPUs (SM560-S / SM560-S-FD-1 / SM560-S-FD-4)

The following communication requirements shall be fulfilled for using AC500-S safety PLC:

- Safety data cannot be transferred over public networks, e.g., internet. If safety data is transferred across company/factory networks, ensure that sufficient protection is provided against manipulation (firewall or router for network separation).
- Equipment connected to communication devices shall feature safe electrical isolation.

> **NOTICE!**
> You can use AC500-S safety I/O modules and SM560-S-FD-1 / SM560-S-FD-4 safety CPUs with 3rd party F-Hosts on PROFINET. Download and install valid ABB GSDML files in your 3rd party F-Host engineering environment from www.abb.com/plc.
>
> After this, you can configure and use these AC500-S modules with 3rd party F-Host. Contact ABB technical support on how to obtain F_iPar_CRC values of AC500-S safety I/O modules for 3rd party F-Hosts.
>
> Validate that all iParameters (input delay, channel configuration, etc.) for all AC500-S safety I/Os and other F-Devices are correct with a given F_iPar_CRC value using appropriate functional validation tests or verification procedure for those parameters *Chapter 6.5 “Verification procedure for safe iParameter setting in AC500-S safety I/Os” on page 341*.
>
> Default F_iPar_CRC values used in GSDML files for AC500-S safety I/O modules do not correspond to default iParameter configurations for AC500-S safety I/O modules and have to be re-calculated in the engineering tools before their usage. It was done to avoid unintended use of AC500-S safety I/O with 3rd party F-Hosts.

2.13 Safety function and fault reaction

The main safety function of AC500-S safety PLC is to read safety digital and analog inputs to control the safety digital outputs by the safety logic module safety CPU according to a user-defined IEC 61131 application program and configuration.

The AC500-S safety PLC can be used as a "de-energize to trip" (normally energized, NE) system. The safe state of the outputs is defined according to the table below:

Table 2: NE safety system behavior

	Normally energized, NE
Mode according to IEC 61508	High-demand or low-demand
Safety function	De-energize to trip
Safe state	De-energized outputs

The purpose of AC500-S safety function is to enable a machine (as a system) to achieve with a given SIL (IEC 61508 and IEC 61511), SILCL (IEC 62061) and PL (ISO 13849-1) a system safe state. An exemplary safety function on the application level, which can be executed by AC500-S in machinery applications, is the emergency stop.

2.13.1 Safety CPU (SM560-S / SM560-S-FD-1 / SM560-S-FD-4)

The safety function of the safety CPU is to correctly process signal information. It processes safety input signals and internal data storage to generate signals to safety output modules and set a new state of its internal data storage.

If this function cannot be correctly executed, the safety CPU goes to a SAFE STOP state, in which no valid safety telegrams are generated and, as a result, all safety output module channels are de-energized (‘0’ state) after watchdog time is expired.

Faults in the cyclic communication between the safety CPU and safety I/O modules or other F-Devices, e.g., SM560-S-FD-1 or SM560-S-FD-4 safety CPUs, are detected by the safety CPU and, as a result, ‘0’ values are handed to the safety application program.

The application program developer must implement a specific fault reaction, e.g., setting safety output channels to de-energized (‘0’ state), when required.

2.13.2 Safety module with safety input channels (DI581-S, DX581-S and AI581-S)

The safety function of safety modules (DI581-S, DX581-S and AI581-S) with digital and analog input channels is to correctly read external analog and/or digital signals. If this function cannot be correctly executed, the safety module or only its input channel, depending on the fault scope, has to go to a safe state. In case of a channel fault, the safe value (de-energized = ‘0’) is transferred to the safety logic module (e.g., SM560-S) with additional information about the fault for the given channel.

In case of module fault, no valid telegrams are generated by the safety module to the safety logic module. The values of those safety input channels will be assigned to safe values (de-energized = ‘0’) on the safety CPU.

Faults in the cyclic communication between the safety CPU and the safety modules are detected by the safety modules with input channels. If a communication fault occurs, all inputs of the affected safety module go to a so-called passivation state in which ‘0’ values are sent as process values when the communication to the safety CPU is re-established. The switch-over (reintegration) from safety values ‘0’ to process data takes place only after user acknowledgment.

2.13.3 Safety module with safety output channels (DX581-S)

The safety function of safety modules (DX581-S) with safety output channels is to correctly write their output channel signals. If this function cannot be correctly executed, the safety module or its output channel group, depending on the fault scope, has to go to a safe state. In case of a channel fault, the safe value (de-energized = ‘0’) is set for the given safety output channels. In case of module fault, no valid telegrams are generated by the safety output module to the safety CPU. The values of all safety output channels will be assigned to safe values (de-energized = ‘0’).

Faults in the cyclic communication between the safety CPU and the safety output modules are detected by the safety output module DX581-S. If a communication fault occurs, all outputs of the affected safety output module are de-energized = ‘0’. The switch-over (reintegration) from safety values ‘0’ to process data takes place only after user acknowledgment, when the communication is re-established.

2.14 Safety function test

After creating a safety program and system configuration, you must carry out a complete function test in accordance with your automation task. For changes made to a safety program which has already undergone a complete function test, only the changes need to be tested, if a proper impact analysis was done before.

Safety application program, safety I/O configuration, etc. have to be verified, printed out and saved for project data report and archive. The system acceptance test shall follow safety function test. After you finish configuring the hardware and assigning parameters for the safety CPU and safety I/O modules, you can perform an acceptance test. During the system acceptance test, all relevant application-specific standards must be adhered.

2.15 Troubleshooting

Error messages in the diagnosis buffer of non-safety CPU include a description, which shall help you to fix potential problems with AC500-S configuration. If some of the problems persist or no error messages are available in the diagnosis buffer, contact ABB technical support for further details.

> **!** **NOTICE!**
> Make sure that safety I/O modules are properly attached to TU582-S terminal unit with a good electrical contact to avoid unintended system state with possibly wrong LED states *Chapter 3.3.3 “Mounting, dimensions and electrical connection” on page 67* *Chapter 3.4.3 “Mounting, dimensions and electrical connection” on page 93* *Chapter 3.5.3 “Mounting, dimensions and electrical connection” on page 113*.

Below you can find a list of known issues and solutions related to AC500-S safety PLC components:

ID	Behavior	Potential cause	Remedy
1.	Safety CPU is in RUN or DEBUG RUN state, but all safety I/O modules suddenly go to RUN (module passivation) state.	Your program may contain endless loop which prevents safety CPU to send valid safety telegrams to safety I/O modules in a proper time manner (within configured watchdog time).	Check (debug) your safety application program and make sure that no endless loop(s) are in your safety application program.
2.	No log in is possible to the safety CPU from CODESYS Safety project.	CODESYS visualization was connected directly to the safety CPU, which blocks the connection to the safety CPU. Only one connection to the safety CPU is allowed at a time.	Disconnect CODESYS visualization from the safety CPU.
3.	During closing or saving of the project, modification of CODESYS Safety project, etc. with Automation Builder, you may see that no reaction comes from the Automation Builder and/or CODESYS Safety project. It is as if the application hangs.	The user management of Automation Builder requires that you confirm your log-on credentials for safety components and issues a log-on message box which is not in the foreground. Your previous log-on session has expired.	Find a log-on message in the background of your Windows desktop, log-on and continue your previous actions. Set longer user log-on session time for Automation Builder if this behavior repeats ➪ *[4]*.
4.	Your safety digital input channel is occasionally passivated with an internal error diagnostic message on non-safety CPU. With AC500 V2 non-safety CPU: error severity: E3, component: 14, device: 1 ... 10, module: 31, channel: 31, error: 43 With AC500 V3 non-safety CPU: error severity: 3, error code: 16171	One of potential reasons is that your input signal frequency exceeded an allowed input channel signal frequency ➪ *allowed frequency ranges: Chapter 3.3.2 on page 64*.	Check that your input signal does not exceed the allowed digital input signal frequency.
5.	DX581-S module is powered on, but no power supply is connected to UP clamps of DX581-S module.	Wiring error on DX581-S module when +24 V DC is connected to at least one of the safety digital output clamps of DX581-S. As a result, DX581-S is powered on through safety digital outputs.	Check the wiring of DX581-S and disconnect +24 V DC from the safety digital output clamp(s).
6.	Some channels of a safety I/O module or a complete safety I/O module is occasionally passivated without a reason (wiring is correct, etc.).	No proper electrical contact between a safety I/O module and TU582-S terminal unit.	Make sure that you pressed the safety I/O module into TU582-S terminal unit with a force of at least 100 N, as prescribed in AC500-S checklists.
7.	With the increased number of safety I/O modules in the system, it takes longer to execute *"Create boot project"* command for the safety CPU.	The safety CPU is a single-threaded system. The more safety I/O modules are in the system, the higher is the internal cycle time of the safety CPU to process safety I/O relevant data.	Currently, there is no possibility to change this behavior other than to split safety I/Os to different safety CPUs, so that each safety CPU has less safety I/Os to handle.
8.	After log in to CODESYS Safety, one can observe a long list of internal constants with a green font color for PROFIsafe F-Host instances.	In CODESYS Safety, the option *"Replace constants"* is selected.	Go in CODESYS Safety menu to *"Project ➔ Options ➔ Build"* Unselect option *"Replace constants"*.

ID	Behavior	Potential cause	Remedy
9.	No valid safety project can be generated (PROFIsafe callback functions are missing and no safety I/O mapping is created).	A potential reason is that you selected in *"Object Properties... ➔ Access rights"* for any of the POUs in the CODESYS Safety project tree the following option: *"No Access"* or *"Read Access"* for all *"User Groups"* with *"Apply to all"* selection.	Start CODESYS Safety project, log in and go to *"Object Properties... ➔ Access rights"* for any of the POUs in the CODESYS Safety project tree to set *"Full access"* for any of user groups followed by selection of *"Apply to all"*. After this, you can successfully repeat *"Create Safety Configuration Data"* command for your CODESYS Safety project from Automation Builder.
10.	I call CurTimeEx FB and always get "0" values on the outputs.	CurTimeEx FB is not implemented in the current version of the safety CPU and is reserved for future use.	Do not use CurTimeEx FB in your safety application program.
11.	● Set "Enable debug" parameter to *"OFF"* on the safety CPU. ● Create boot projects for the safety CPU and the non-safety CPU. ● Execute a power cycle. ● Go to CODESYS Safety and compare boot project CRCs on your PC and the safety CPU. The comparison shows that they are the same, which is OK. ● Try to create a boot project for the safety CPU. An error message follows because of "Enable debug" set to *"OFF"* for the safety CPU, which is OK. ● Repeat the comparison of boot project CRCs on your PC and the safety CPU. They are reported to be not equal now (boot project CRC for the safety CPU is shown as CDCDCDCD), which can be misleading since the boot project on the safety CPU was not changed.	CODESYS Safety does not support the described use case.	After power cycle of safety CPU, the correct boot project CRC shall be shown for the safety CPU.
12.	The serial driver is used to connect to safety CPU. One executes *"Login"* command shortly followed by *"Logout"* command in CODESYS Safety and shortly after this the *"Login"* command is again executed. After second log in attempt, the communication error is shown in CODESYS Safety.	The serial driver does not have enough time to be re-initialized.	Wait for at least 20 seconds before executing *"Login"* command after *"Logout"* was performed.

ID	Behavior	Potential cause	Remedy
13.	● One executes *“Login”* command in CODESYS Safety and uses *“setpwd”* PLC browser command to set a new password, e.g., "PWD1" for the safety CPU. ● Power cycle is executed for the safety CPU, but CODESYS Safety window remains open on end-user PC. ● One executes *“Login”* command in CODESYS Safety and enters the new password "PWD1", which was set in Step 1. One uses *“setpwd”* PLC browser command to set a new password, e.g., PWD2, for the safety CPU. ● Power cycle is executed for the safety CPU, but CODESYS Safety window remains open on end-user PC. ● One executes *“Login”* command in CODESYS Safety and an error message is shown "You have entered a wrong PLC password!". After pressing *“OK”*, you still have a possibility to enter a new password "PWD2" and successfully log in to safety CPU.	CODESYS Safety instance attempts to log in to the safety CPU with an old password.	After resetting the safety CPU password, close CODESYS Safety instance and open it again. The error message will not appear again.
14.	After power-on, safety I/O module goes to SAFE STOP state with both ERR LEDs = ON.	The configured F_Dest_Add value in Automation Builder project is not equal to the PROFIsafe address switch value on the safety I/O module.	Make sure that F_Dest_Add value in Automation Builder project is equal to the PROFIsafe address switch value on the safety I/O module.
15.	No log in to the safety CPU is possible.	Wrong "Communication parameters..." settings are used.	Check that correct settings of "Communication parameters..." are used in CODESYS Safety to connect to the safety CPU.
16.	After the boot project is loaded to the safety CPU, sometimes the V2 non-safety CPU seems to do nothing for about 45 seconds until its ERR-LED is switched on.	Timeout in V2 non-safety CPU.	Such situation can be observed very seldom. There is no remedy for this behavior of V2 non-safety CPU at the moment.
17.	After power-on of the safety CPU, it may happen that the safety CPU does not go to RUN mode. DIAG LED is ON and no boot project is loaded to the safety CPU. If you attempt to log in to the safety CPU, then the following error message can be seen in CODESYS Safety: "No program on the controller! Download the new program?".	Safety CPU power dip function is triggered if the pause between the power-off and the following power-on phase is less than 1.5 s. The boot project is still on the safety CPU, but not loaded due to power dip detection. Thus, there is no need to reload any boot project to the safety CPU.	Do power-off and power-on of the safety CPU with a pause between power-off and power-on phase ≥ 1.5 s.

ID	Behavior	Potential cause	Remedy
18.	If a breakpoint is reached in CODESYS Safety during debugging and you try to force a variable, then this variable is updated with the forced value only in the next safety CPU cycle.	The safety CPU is single-threaded.	This behavior is as designed.
19.	During project download to the safety CPU, the download window stays with 0 bytes of downloaded code forever or an error message pops up.	"Enable debug" parameter was set to *“OFF”* for the safety CPU and this configuration data was downloaded to non-safety CPU.	Set "Enable debug" parameter to *“ON”*, generate a new configuration and download project to non-safety CPU. New project code can be now downloaded to the safety CPU through CODESYS Safety.
20.	Unable to log in to the safety CPU from CODESYS Safety after logout.	Too fast log in to the safety CPU after logout in CODESYS Safety.	Wait a few seconds (~ 5 - 10 s) after logout from the safety CPU before you perform log in to the safety CPU in CODESYS Safety.
21.	Diagnosis message with error severity level 3 and error text "Measurement underflow at the I/O module" appears in non-safety CPU diagnosis system despite the fact that overcurrent and not undercurrent was observed for the given AI581-S input channel.	The internal detection mechanism is not always able to differentiate between over- and undercurrent because the overcurrent is often followed by undercurrent effects in AI581-S electronics.	There is no remedy for this problem yet.
22.	"Enable debug" parameter = *“ON”* was set for the safety CPU and correctly loaded to non-safety CPU. However, one still cannot debug on the safety CPU.	CODESYS Safety projects on your PC and in the safety CPU are not the same. You may get also the following message window with the text: "The program has changed! Download the new program?".	Download your CODESYS Safety project from your PC to the safety CPU and debugging shall be possible now.
23.	After using CODESYS Safety menu item *“Online ➔ Reset”*, the safety CPU goes to DEBUG STOP state (non-safety mode). Safety I/O modules go to module passivation state. If you log in to the safety CPU, then you can see OA_Req_S = TRUE bits in PROFIsafe instances of F-Devices. The safety application is not executed by the safety CPU, but you still can set OA_C = TRUE for F-Devices and they will go to RUN mode. The safety CPU remains in DEBUG STOP state (non-safety) all the time.	PROFIsafe F-Host does not run in fail-safe mode after using CODESYS Safety menu item *“Online ➔ Reset”*.	This behavior is as designed in the safety CPU *Chapter 3.1.5.1 “Description of safety CPU module states” on page 45*.
24.	Error message "Error in configuration data, safety PLC cannot read configuration data" is available on the safety CPU.	● Downloaded configurations on non-safety CPU and safety CPU do not fit to each other. ● No boot project is loaded on the safety CPU.	● Download valid configurations, as part of boot projects, on non-safety CPU and the safety CPU, respectively, and make sure that they fit to each other. ● Download a valid boot project on the safety CPU.

2.16 FAQ - AC500-S safety PLC

- **Boot project availability on the safety CPU after power dip or incomplete power cycle**

  In case of an under- or overvoltage, which may be also caused by an incomplete power cycle (power-off followed by power-on in less than 1.5 s), the safety CPU goes to SAFE STOP state with I-ERR LED ON. However, the boot project is still intact. To put the safety CPU back to RUN mode, it is necessary to perform two subsequent power cycles. After the first power cycle, the safety CPU goes to DEBUG STOP (non-safety) mode state with DIAG LED ON. The second power cycle puts the safety CPU back to RUN (safety) mode.
- **Not possible to create a boot project for the safety CPU**

  Check if the parameter "Enable Debug" for the safety CPU is set to "ON" in Automation Builder project and the generated boot project was loaded to the non-safety CPU followed by a power cycle.
- **After power cycle, the safety CPU goes into SAFE STOP state (I-ERR ON)**

  This situation could arise due to a corrupt boot project or the rotary switch setting in the safety CPU is wrongly set to one of these values: 0xFE, 0xFD or 0xFC. Another possibility is that the safety CPU was powered off too short. To ensure a reliable restart the power-off time must be > 1.5 s.).
- **Channel reintegration of AI581-S safety module is not possible after removal of the fault condition**

  Only in the case of a channel passivation due to overcurrent or undercurrent the safety analog channel remains passivated for 30 s to restore its initial properties and then the check is performed if the error condition is still present or not. If the error has gone, then the reintegration request signal for the given channel is set to TRUE to allow channel reintegration. Within previously mentioned 30 s time, the safety analog channel cannot be reintegrated.
- **Process value of certain configured input is always FALSE (only in 2-channel evaluation mode)**

  Our modules are designed in such a way that, in a 2-channel mode, the lower channel (e.g., channels 0/4 ➔ Channel 0, channels 1/5 ➔ Channel 1, etc. for DX581-S module) always transports the aggregated process value, PROFIsafe diagnostic bit, acknowledgment request and acknowledge reintegration information. The higher channel always provides the passivated value "0". Thus, a name mapping for the higher channel is not required in a 2-channel evaluation mode.
- **Acyclic non-safe data exchange takes a very long time**

  This behavior depends on the task configuration setting in your non-safety CPU. Adjust the cycle time (e.g., set task cycle time to 1 ms) of your task on non-safety CPU where the acyclic non-safe data exchange FBs are programmed to obtain the best performance.
- **When should I use cyclic non-safe data exchange instead of acyclic non-safe data exchange?**

  If 84 bytes in acyclic non-safe data exchange are not enough or data exchange is too slow, you can use cyclic non-safe data exchange for data up to 2 kB with minimum programming effort.

  In most safety applications, this functionality is not needed and, thus, shall not be used. However, if you still need it, refer to *Appendix B.5 “Data exchange between safety CPU and AC500 V2 non-safety CPU” on page 384* *Appendix C.5 “Data exchange between safety CPU and AC500 V3 non-safety CPU” on page 400*.
- **Is data communication using acyclic or cyclic non-safe data exchange safe?**

  Data communication using acyclic or cyclic non-safe data exchange is non-safe, because it is not protected by any functional safety measures for data communication. However, customers may implement their own safety profiles on top of this non-safe communication using so-called "black channel" principle. Contact ABB technical support for details.
- **How discrepancy time handling (2-channel configuration) in S-DIs is implemented?**

  To get a robust evaluation of a 2 channel configuration, it's very important to handle noises, disturbances and other influences at safety digital inputs in a reliable way. Without such a behavior, a flickering on a channel would cause a faulty 2 channel evaluation of the given process value, which would jeopardize system availability.

- **No detection of wire cross-talk or short circuit to 24 V DC for S-DOs of DX581-S. Why and how to solve this problem?**

  The outputs of the DX581-S safety module are decoupled from the connected load. This is necessary to avoid any influence of connected load on the internal test circuit and, thus, guarantee high robustness (no occasional trips due to false error detection caused by unexpected change of electrical characteristics of the connected load). Therefore, wire cross-talk and short circuit to 24 V DC can be detected only up to the output clamp of DX581-S safety output, but not on the attached output wire. In most customer cases, error exclusion due to output wire isolation or, alternatively, the machine re-start (with proper start-up test procedure implemented in the safety CPU program for given S-DOs to activate them one after each other) at least 1 per month is often enough. The user may also take other appropriate actions (e.g., by defining appropriate test periods for the safety function or by reading back the status of the output wire using a safety digital input) to satisfy their respective IEC 62061 and ISO 13849-1 requirements, if wire cross-talk or short-circuit to 24 V DC shall be implemented.
- **Is my safety program OK if not all safety programming guidelines and rules checked by AC500-S safety code analysis (SCA) rules are satisfied?**

  SCA tool only checks whether the static safety programming guidelines or rules are followed. As such, any errors identified by SCA tool may not necessarily result in machine malfunction but will require additional argumentation why those exceptions (not fulfilled safety programming guidelines or rules) are allowed in the given customer safety application case. The latter may delay the certification of customer safety application program.
- **What does built-in power supply in the safety I/O module mean?**

  It means that no separate power supply module shall be bought for AC500-S safety I/Os. 24 V DC can be directly connected through UP and ZP pins on the terminal unit.
- **What is the effect of connecting test pulse of the same type (e.g., T0, T1, T2, T3, etc.) from one module to the safety digital input channel of another module? Are test pulses module-specific?**

  Yes, test pulses are module-specific. As test pulses are module-specific, connecting any test pulse of the same type from one module and still the same channel on the other module would cause channel passivation. This kind of connection is not permitted and not recommended.
- **Will there be a different delay of safety telegram if the safety module is placed in another physical slot (communication module or I/O module slot)?**

  The telegram delay difference can be negligible in such cases and possible difference is far below 1 ms.
- **Is 1oo2 internal safety structure applicable for safety inputs only when we have 2-channel input?**

  No, the entire AC500-S hardware system is designed using 1oo2 internal safety structure. Hence, even when you connect a single input, internally it is split and processed using 1oo2 safety architecture.
- **How to interface safety mats/bumpers and safety edges?**

  Most of the safety mats and bumpers in the market come with ASi-Safety option. With the help of ASi-Safety to PROFINET/PROFIsafe gateway, you can connect such signals to AC500-S.
- **Can we use 2-wire transmitters with analog input?**

  Yes, AI581-S analog module is equipped to handle both 2-wire and 4-wire transmitters.
- **What is the ON time of a test pulse in DI581-S/DX581-S modules? How often is it repeated?**

  Test pulse terminal clamps provide 24 V DC signal for monitoring passive sensors with test pulses. This test pulse signal is switched off for a fixed time (1 ms) to LOW state. This is valid for both DI581-S and DX581-S module. The test pulse repeats every 58 ms for DI581-S and every 27 ms for DX581-S module on each test pulse channel.
- **How often is the safety output OFF when the detection feature is made ON in DX581-S module?**

  If the detection is enabled, the output of the DX581-S safety module is tested every 55 ms. Be aware, that the test pulse of the internal main switch can also be observed on each output. The main switch test pulse cannot be disabled and is always present. Its duration is slightly below 1 ms in the worst-case (if the output current is 500 mA) and is almost not visible in the best-case (if the output current is below 50 mA).

- **Can AC500-S safety modules be used in low-demand applications?**
  Yes.
- **How to make the safety CPU address switch setting compliant to SIL 3 / PL e if one wants to use its value in the safety application program?**
  One may want to change the safety CPU safety program execution path depending on the safety CPU configuration switch setting, which can be read in the safety program using SF_SM5XX_OWN_ADR function block. Changing the safety CPU safety program execution path depending on the safety CPU address switch setting only is not always enough to reach SIL 3 / PL e. One has to implement some additional mechanisms, e.g., to have a second point-of-entry for program configuration setting on the application level. This can be done, e.g., by reading some pre-configured (pre-saved) values from SD card on the non-safety CPU. This additional pre-configured (pre-saved) value has to be transferred to the safety CPU and compared against the safety CPU address switch setting before the safety CPU address switch setting is accepted for the safety CPU safety program execution path change. This way one can attain a higher functional safety level up to SIL 3 / PL e.
- **In which types of applications are FBs like SF_APPL_MEASURE_BEGIN and SF_APPL_MEASURE_END used?**
  These FBs can be used for time profiling of your safety application program, which is often very useful for debugging purposes to find performance bottle-neck in safety applications. For instance, to estimate the actual time taken by the safety CPU to execute a certain part of the safety program logic.
- **How can user data on the safety CPU be made persistent?**
  User data can be stored in the non-volatile flash memory of the safety CPU and read or deleted from there using special FBs (SF_FLASH_WRITE, SF_FLASH_READ and SF_FLASH_DEL).
- **Can errors related to remote PROFINET/PROFIsafe safety modules be captured in the diagnostic buffer of the non-safety CPU?**
  With AC500 V2 non-safety CPU:
  Yes, you can use special diagnostic FBs to read diagnostic messages from remote safety modules on the V2 non-safety CPU. These FBs can be found in the library Profinet_AC500_V13.lib on the V2 non-safety CPU.
  With AC500 V3 non-safety CPU:
  The PROFINET/PROFIsafe related errors can be automatically collected in the diagnostic buffer of the V3 non-safety CPU.
- **Why does non-safety CPU reboot command not reboot remote safety I/O modules?**
  This behavior is as designed. Only central safety I/O modules will be re-initialized after non-safety CPU reboot command. All remote safety I/O modules may not be re-initialized and have to be acknowledged from the safety program to re-integrate them after non-safety CPU and safety CPU re-initialization is finished. This behavior (re-initialization or not) depends on PROFINET CI5xx setting and can be modified.
- **Is ST to LAD/FBD conversion possible?**
  Yes, for simple projects involving basic instruction set the conversion is possible. However, not all standard ST constructs can be converted to LAD/FBD. Please keep in mind that after a conversion from ST to LAD/FBD you cannot reverse the safety program code back to ST.
- **In antivalent mode wiring, the NO channel is always connected to the lower channel (the channel that delivers an aggregated 2-channel safety value to the safety CPU). Is there any specific reason for this?**
  This behavior is as designed to avoid any faults during antivalent sensor wiring and potential misinterpretation of which channel delivers an aggregated 2-channel safety value.
- **While using our safety and non-safety I/Os with 3rd party safety (F-Host) and non-safety CPU, will safety and non-safety I/O diagnostic messages be available in the diagnostic buffer of those 3rd party safety (F-Host) and non-safety CPUs?**
  All diagnostic messages from safety and non-safety I/Os are non-safe data which is collected by non-safety CPU (also 3rd party one). All diagnostic messages from safety and non-safety I/Os are currently available in AC500 diagnostic message format and can be read and put in the diagnostic buffer of 3rd party non-safety CPU by invoking special FBs or using standard PROFINET diagnosis.
- **Who could certify a safety program?**
  All international and national accredited certification bodies like TÜV, EXIDA, UL, etc. (some of them operating around the world) could certify a safety program.

- **What are the right steps to develop a safety program?**

  You have to refer to ISO 13849-1 and IEC 62061 guidelines for machine safety application development and to IEC 61511 for process safety application development.
- **Is it allowed to use FOR loops as an alternative for IF and CASE for boundary checks in arrays?**

  No, it is not allowed to use it as an alternative.

  If arrays are used in FOR loops, the programmer must still implement boundary checks.

3 AC500-S safety modules

3.1 Safety CPU - SM560-S / SM560-S-FD-1 / SM560-S-FD-4

Elements of the module

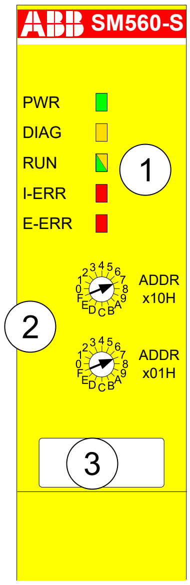


Fig. 5: SM560-S / SM560-S-FD-1 / SM560-S-FD-4

1 Five LEDs for status display
2 Rotary switch for address/configuration setting
3 Label

3.1.1 Purpose

SM560-S / SM560-S-FD-1 / SM560-S-FD-4 are safety CPUs for up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1) safety applications. The safety CPU is mounted on the left side of the non-safety CPU on the same terminal base. The communication between the non-safety CPU and the safety CPU takes place through the internal communication bus, which is integrated in the terminal base.

Depending on the used terminal base and used non-safety CPU, more than one communication module can be simultaneously employed at one non-safety CPU. However, only one safety CPU can be operated simultaneously at one non-safety CPU.

The safety CPU is programmed and configured via the DPRAM using safety system configurator and CODESYS Safety programming environment, which are a part of the Automation Builder software.

The configuration of the safety CPU is saved non-volatile in its flash EPROMs.

Information on how to combine safety CPUs with its non-safety environment can be found in the compatibility information *Appendix B.1 “Compatibility with AC500 V2 non-safety CPU” on page 373* *Appendix C.1 “Compatibility with AC500 V3 non-safety CPUs” on page 391*.

3.1.2 Functionality

3.1.2.1 Overview

AC500 safety CPUs are always used with non-safety CPUs.

Programming of the safety CPU is done using CODESYS Safety in a similar way as programming of AC500 CPU, but in accordance with the guidelines ↯ *[1]*. Programming is done by means of routing via the AC500 CPU using the serial interface or Ethernet. The user program is composed of:

- Compiled code of all POUs called in the program
- Initialization code for variables

SM560-S-FD-1 / SM560-S-FD-4 contains all features of SM560-S safety CPU. Additional features available on SM560-S-FD-1 / SM560-S-FD-4 safety CPUs are:

- PROFIsafe F-Device functionality
  - SM560-S-FD-1 (-XC) is able to communicate with 1 PROFIsafe F-Host (controller)
  - SM560-S-FD-4 (-XC) is able to communicate with up to 4 PROFIsafe F-Hosts (controllers)
- Bigger safety program size: 1.3 MB (SM560-S safety CPU has 1.0 MB)

Each safety CPU variant has its own product identifier in the production data. Thus, a download of a boot project to a wrong product variant is detected by its firmware.

3.1.2.2 Floating-point operations

Safety CPUs can perform floating-point operations.

DANGER!
Divisions by zero are not allowed and shall be caught latest during the formal safety CPU code review according to CODESYS Safety programming guidelines ↯ *Chapter 4.4 “CODESYS Safety programming guidelines” on page 179*.

If case of exceptions during floating-point operations (e.g., due to usage of invalid arguments), the safety CPU goes to a SAFE STOP state or delivers a return value "Infinity".

Note that the range of valid arguments in safety CPU for floating-point functions is:

- SIN and COS: [-9×10^{15} ... 9×10^{15}]
- TAN: [-4.5×10^{15} ... 4.5×10^{15}]
- ATAN: [-3.402823×10^{38} ... 3.402823×10^{38}]
- LOG, LN and SQRT: up to 3.402823×10^{38}

The arguments outside the above-presented range will lead to a SAFE STOP state of the safety CPU.

DANGER!
The end-result of floating-point operation has to be checked for its validity before it is further used in the safety program.

DANGER!
It is important to take into account the following while programming with floating-point arithmetic ➡ *[6]*:

– Round or truncate results after each floating-point operation according to defined ULPs (MOD, EXPT, EXP, ABS, TAN, ASIN, ACOS, ATAN, SIN, COS, LOG and LN operations are executed with a maximum expected error of 2 ULP; ADD, SUB, MUL, DIV and SQRT are executed with a maximum error of 1 ULP in the safety CPU). *http://en.wikipedia.org/wiki/Unit_in_the_last_place* for more details on ULPs.
– If you compute a value which is the result of a sequence of floating-point operations, the error can accumulate and greatly affect the computation itself.
– Whenever subtracting two numbers with the same signs or adding two numbers with different signs, the accuracy of the result may be less than the precision available in the floating-point format.
– The order of evaluation can affect the accuracy of the result.
– When performing a chain of calculations involving addition, subtraction, multiplication and division, try to perform the multiplication and division operations first.
– When multiplying and dividing sets of numbers, try to arrange the multiplications so that they multiply large and small numbers together; likewise, try to divide numbers that have the same relative magnitudes.
– When comparing two floating-point numbers, always compare one value to see if it is in the range given by the second value plus or minus some small error value.

3.1.2.3 System functions

The safety CPU is not equipped with a battery. Therefore, all operands are initialized once the control voltage is switched on. Data exchange between safety and non-safety CPUs is possible ➡ *Appendix B.5 “Data exchange between safety CPU and AC500 V2 non-safety CPU” on page 384* ➡ *Appendix C.5 “Data exchange between safety CPU and AC500 V3 non-safety CPU” on page 400*.

DANGER!
It is not recommended to transfer data values from non-safety CPU to safety CPU. If doing so, end-users have to define additional process-specific validation procedures in the safety program to check the correctness of the transferred non-safety data, if they would like to use those non-safety values for safety functions.

It is of no concern to transfer data values from safety CPU to non-safety CPU, e.g., for diagnosis and later visualization on operator panels.

Self-tests and diagnostic functions (both start-up and runtime), like CPU and RAM tests, program flow control, etc. are implemented in the safety CPU according to IEC 61508 requirements.

Selected data can be stored fail-safe and permanently in the flash memory of the safety CPU using special library POUs SF_FLASH_READ, SF_FLASH_WRITE and SF_FLASH_DEL ➡ *Chapter 4.6.7.10 “SF_FLASH_READ” on page 314* ➡ *Chapter 4.6.7.11 “SF_FLASH_WRITE” on page 316* ➡ *Chapter 4.6.7.12 “SF_FLASH_DEL” on page 319*.

The safety CPU is a single threaded and single task CPU. Only one free-wheeling program task is available for safety program execution. The free-wheeling task is the task which will be processed as soon as the safety program is started and at the end of one run will be automatically restarted in a continuous loop. There is no cycle time defined for this task, but users can supervise the cycle time of the safety CPU using a special library POU SF_WDOG_TIME_SET *Chapter 4.6.7.3 “SF_WDOG_TIME_SET” on page 309*.

The watchdog time of the safety CPU set using SF_WDOG_TIME_SET is the maximum permissible time allowed for its cycle time run. If the time set in SF_WDOG_TIME_SET is exceeded during the program execution on the safety CPU, then it goes to a SAFE STOP state (no valid telegrams are generated by the device) with I-ERR LED = ON.

> **NOTICE!**
> POU SF_WDOG_TIME_SET must be called in the user program only one time to set some watchdog value greater than 0. If SF_WDOG_TIME_SET is not called in the user application program, the default watchdog time = 0 is used, which leads the safety CPU directly to a SAFE STOP state with I-ERR LED = ON.
>
> To avoid occasional stops of the safety CPU due to cycle time overrun detected by the cycle time monitoring, one shall observe the safety CPU load in the test run of the user application program to make sure that the selected watchdog monitoring value was correctly set.

> **NOTICE!**
> The watchdog value set in POU SF_WDOG_TIME_SET is used for the safety CPU cycle time monitoring only in RUN (safety) mode. In DEBUG RUN (non-safety) and DEBUG STOP (non-safety) modes of the safety CPU, the watchdog value is ignored.

Using a special PLC browser command "setpwd", it is possible to set a password for the safety CPU to prevent an unauthorized access to its data (application project, etc.). Without knowledge of this password, no connection to the safety PLC can be established.

3.1.2.4 Power supply supervision

The internal power supply (+3.3 V) of the safety CPU is supervised for under- and overvoltage. In case of under- or overvoltage is detected, the safety CPU goes to a SAFE STOP state (no valid telegrams are generated by the device) with I-ERR LED = ON. To avoid continuous restart of the safety CPU after power supply is back within an allowed voltage range, one can set the maximum allowed number of the safety CPU restarts using POU SF_MAX_POWER_DIP_SET *Chapter 4.6.7.2 “SF_MAX_POWER_DIP_SET” on page 308*. As soon as the maximum allowed number of the safety CPU restarts is exceeded, the safety CPU does not restart and remains in the SAFE STOP state until the user explicitly executes a power cycle.

3.1.2.5 Address / configuration switch / F_Dest_Add settings

The setting of two rotary switches for PROFIsafe address and/or system configuration (for example, these switches can be used for safety program flow control) can be read out in the safety application program using POU SF_SM5XX_OWN_ADR *Chapter 4.6.7.8 “SF_SM5XX_OWN_ADR” on page 313*. Switch address values 0xFF, 0xFE, 0xFD and 0xFC are used for internal safety CPU system functions described below:

- Switch address value 0xFF during the start of the safety CPU prevents loading the boot project to the safety CPU on start-up (the boot project still remains in the flash memory of the safety CPU). As a result, the user is able to log-in to the safety CPU and load a new correct boot project. This can be needed if the boot project is corrupt and could lead to a SAFE STOP state of the safety CPU. The safety CPU goes to DEBUG STOP (non-safety) state after start-up and successful 0xFF command execution.
- Switch address value 0xFE during the start of the safety CPU allows deleting the boot project from its flash memory. The boot project is finally deleted after a power cycle of the safety CPU. This can be needed if the boot project is corrupt and could lead to a SAFE STOP state of the safety CPU. The safety CPU goes to SAFE STOP state after start-up and 0xFE command execution.
- Switch address value 0xFD during the start of the safety CPU allows deleting user data from its flash memory. The user data are finally deleted after a power cycle of the safety CPU. This can be needed if user data are corrupt and could lead to a SAFE STOP state of the safety CPU. The safety CPU goes to SAFE STOP state after start-up and 0xFD command execution.
- Switch address value 0xFC during the start of the safety CPU allows deleting all safety CPU data, which includes, in addition to boot project and user data, also safety CPU password and defined power dip value from the flash memory. This means that the safety CPU will be brought to its original state. The data is finally deleted after a power cycle of the safety CPU. The safety CPU goes to SAFE STOP state after start-up and 0xFC command execution.

The switch address value range 0xF0 ... 0xFB is reserved for future internal system functions.

> **!** **NOTICE!**
>
> Usage of switch address values from the system range 0xF0 ... 0xFF can lead to the loss of important user information in the flash memory of the safety CPU, e.g., boot project, user data, password or power dip value can be lost. Therefore, it is important that users pay a special attention during the change of switch address position on the safety CPU.

DANGER!

Despite the fact that SF_SM5XX_OWN_ADR function is a safety POU, the hardware switch address value is a non-safety value and needs additional measures to satisfy functional safety requirements.

PROFIsafe F_Dest_Add addresses for F-Devices on SM560-S-FD-1 / SM560-S-FD-4 safety CPUs are defined using the rotary address switch. It means that the rotary address switch on safety CPUs can have more than one function behind. This shall be carefully considered during the safety application design, for example, if system functions (0xFF, 0xFE, 0xFD and 0xFC values on the rotary address switch) have to be used on SM560-S-FD-1 / SM560-S-FD-4 safety CPUs. In the latter case, the previously defined rotary address switch value for F_Dest_Add addresses shall be properly documented and set back to its original documented value after system functions on the safety CPU were successfully performed.

Usage of the rotary address switch for F_Dest_Add setting allows using the same safety CPU boot project for different machines provided that each machine will have a unique pre-set F_Dest_Add address defined with the rotary address switch and properly engineered in Automation Builder project.

The allowed range of the rotary address switch value for F_Dest_Add setting is 1 to 239 (0 would indicate no usage of F-Devices on SM560-S-FD-1 / SM560-S-FD-4). One rotary address switch represents F_Dest_Add for all possible F-Device instances (maximum 32 F-Device instances each with 12 bytes of safety data) on SM560-S-FD-1 / SM560-S-FD-4 safety CPUs.

The following rule applies for F_Dest_Add assignment to F-Devices:

– F_Dest_Add for F-Device = Rotary address switch value * 100 + F-Device instance number (0..31, which is the consecutive number as F-Devices are instantiated in Automation Builder module/device tree).
– To properly configure F-Device on SM560-S-FD-1 and SM560-S-FD-4 safety CPUs, one has to provide the correct configuration of F_Dest_Add using the rotary address switch value and F- Parameter configuration provided from F-Host and its controller.

A complex system containing multiple AC500-S sub-systems connected together via PROFIsafe needs some additional consideration on how to allocate F_Dest_Add and F_Source_Add addresses because messages from different F-Hosts can overlap in the "Black Channel", for example in non-safety CPU. The potential overlapping may increase the probability of dangerous error in the safety configuration and communication. The typical PFH value for PROFIsafe communication is 3.0E-10.

DANGER!

For each AC500-S sub-system, which PROFIsafe communication can overlap in the "Black Channel" with the PROFIsafe communication from another F-Host, a pair of F_Dest_Add and F_Source_Add (so-called codename in PROFIsafe terminology ⇘ *[3]*) have to be unique. If only F_Dest_Add is checked by the F-Device (e.g., using hardware address settings on it), then not only codenames but also F_Dest_Add shall be unique. In case of SM560-S-FD-1 and SM560-S-FD-4, due to the fact that PROFIsafe communication from different F-Hosts (PROFIsafe telegrams from own F-Host on SM560-S-FD-1 or SM560-S-FD-4 and PROFIsafe telegrams from external F-Hosts) will overlap on non-safety CPU, additional measures to unique codenames shall be applied:

– Unique F_Dest_Add for all F-Devices belonging to external F-Host(s) and own F-Host on SM560-S-FD-1 or SM560-S-FD-4 safety CPUs.

> **NOTICE!**
> Only F_Dest_Add is used for PROFIsafe F-Device identification in SM560-S-FD-1 and SM560-S-FD-4.

The allowed range for F_Dest_Add addresses is described in *Chapter 4.3.5 "Instantiation and configuration of safety modules / definition of variable names" on page 139*.

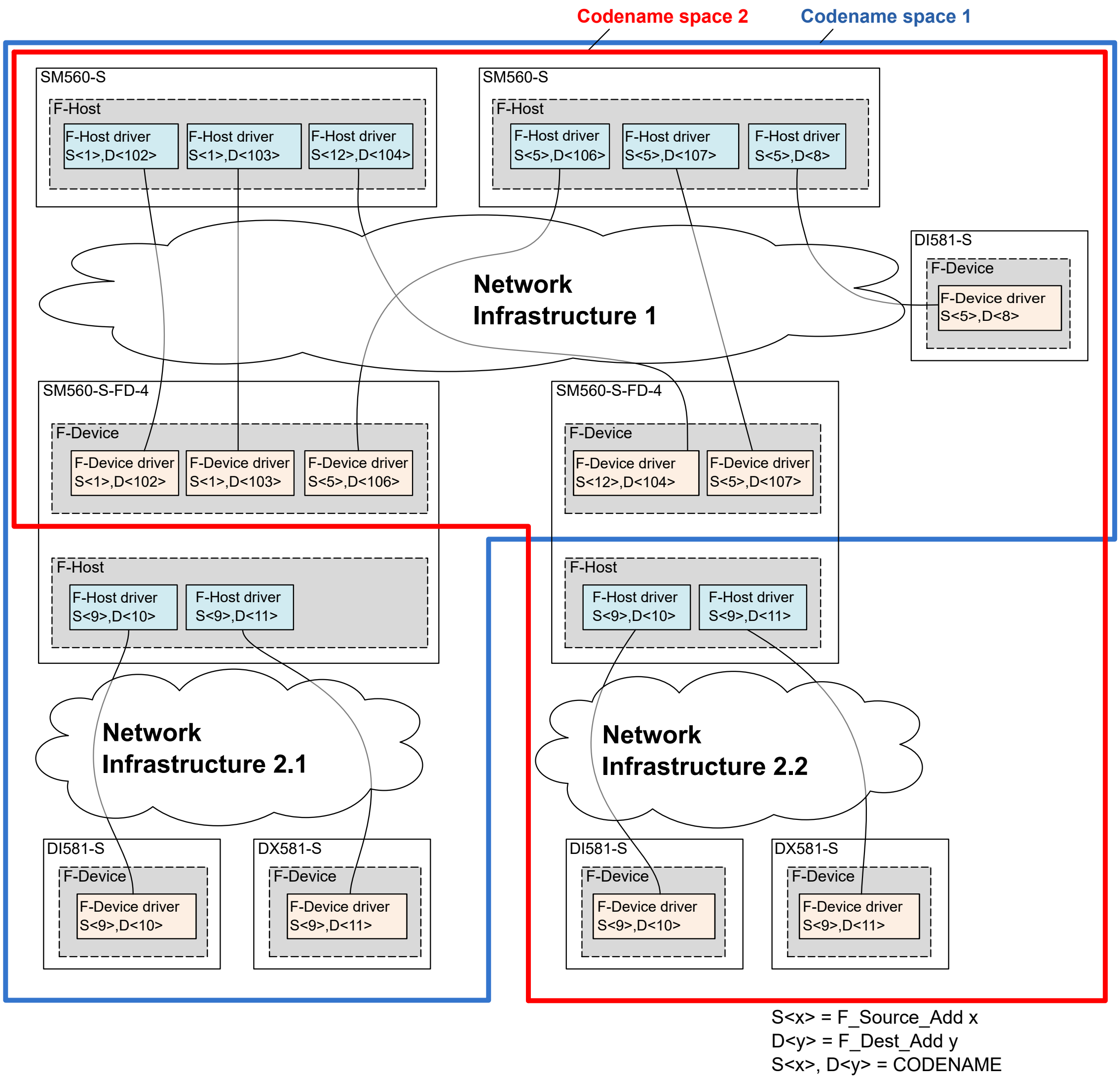


Fig. 6: Exemplary system with overlapping PROFIsafe networks and PROFIsafe address allocation and generic network infrastructure, which may include WLAN, switched network, direct connection, etc.

DANGER!
As a summary, the following rules shall be applied using organizational procedures for safe CPU to CPU communication using SM560-S-FD-1 and SM560-S-FD-4 CPUs (This has to be checked manually and is a part of *Chapter 6.3 “Checklist for configuration and wiring” on page 337*:

- In the same codename space, F_Dest_Add shall be unique (Fig. 6 on page 39).
- In the same codename space, F_Source_Add shall not be re-used in other F-Hosts. Inside the same F-Host, a re-use is allowed for several F-Host drivers.
- In the same codename space, F_Dest_Add shall not be used as F_Source_Add and vice versa.

To ensure that the right safety configuration and safety application is loaded to the right system, customers can use SM560-S-FD-1 / SM560-S-FD-4 address switch to verify that the configuration fits to the selected system. The address switch on SM560-S-FD-1 / SM560-S-FD-4 implicitly protects the given safety CPU because it is used for the definition of F_Dest_Add for PROFIsafe F-Device instances. If a wrong boot project is loaded on the given SM560-S-FD-1 / SM560-S-FD-4, then it will not match to F- Parameters transferred from the F-Host and will end in the configuration error of the corresponding PROFIsafe instance.

3.1.2.6 Firmware, boot code and boot project update

The updates of the safety CPU for boot project, firmware and boot code are performed via non-safety CPU, either via Automation Builder or via SD card.

DANGER!
Each firmware and boot code update has to be followed by a complete functional safety validation procedure for a given safety process control application.

3.1.2.6.1 Update via Automation Builder

We recommend to update firmware, boot code and boot project via Automation Builder. This feature is described in *[4]*.

3.1.2.6.2 Update via SD card

DANGER!
If you use an SD card for firmware, boot code and boot project update via non-safety CPU, it is important that special organizational procedures (e.g., limited access to the cabinet where safety CPU is located) on the end-customer site are defined to avoid unintended software update on the safety CPU.

It is possible to read the current firmware version of the safety CPU using POU SF_RTS_INFO *Chapter 4.6.7.9 “SF_RTS_INFO” on page 313*. Thus, you can limit safety program execution only to the pre-defined firmware versions.

> **NOTICE!**
> You can not update both boot project and firmware / boot code at the same time for the safety CPU. Perform these updates in two steps. It means that you may need two SD cards. One SD card with firmware / boot code update and the other SD card with boot project update.

Firmware and boot code update

The procedure for SD card creation with firmware / boot code for a safety CPU is handled in the same way as described for communciation modules in ⇨ *[4]*.

Boot project update

The safety CPU boot project can be updated only if no boot project is present on the safety CPU. This is to avoid unintentional boot project update on the safety CPU. Before updating a new boot project, delete the existing boot project on the safety CPU, e.g., via setting the address switch to value 0xFE/0xFC ⇨ *Chapter 3.1.2.5 “Address / configuration switch / F_Dest_Add settings ” on page 37*, via PLC browser command `delappl` in CODESYS Safety or *“Online ➔ Reset Origin”* in Automation Builder.

3.1.3 Mounting, dimensions and electrical connection

The safety CPU is mounted on the left side of the non-safety CPU on the same terminal base. The electrical connection is established automatically when mounting the safety CPU. Basic information on system assembly is shown here. Detailed information can be found in ⇨ *[4]*.

Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204 part 1, by skilled electricians only.

Assembly of the safety CPU

> **DANGER!**
> Hot plug and hot swap of energized modules is not permitted. All power sources (supply and process voltages) must be switched off while working with safety modules.

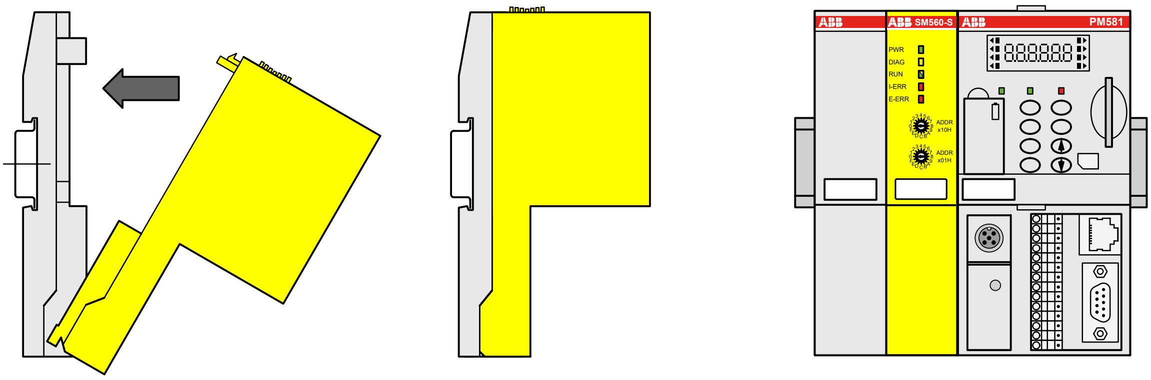


Fig. 7: Assembly instructions

▷ Insert the module below, and then click-in above.

Disassembly of the safety CPU

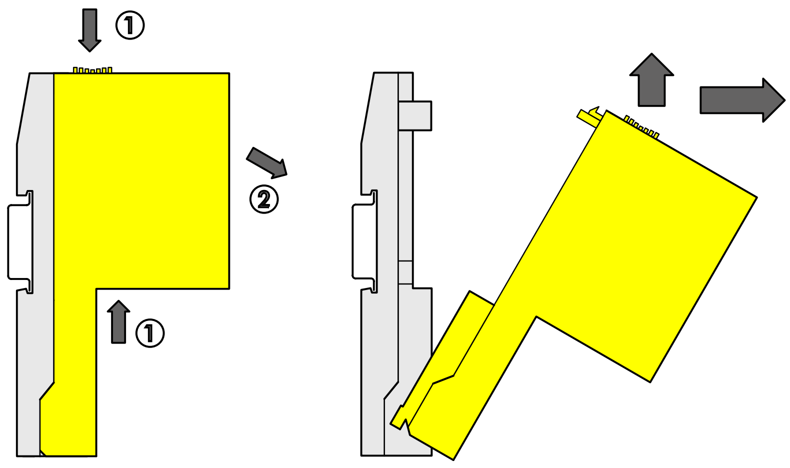


Fig. 8: Disassembly instructions

▷ Press above and below, then swing out the module and remove it.

Dimensions of the safety CPU

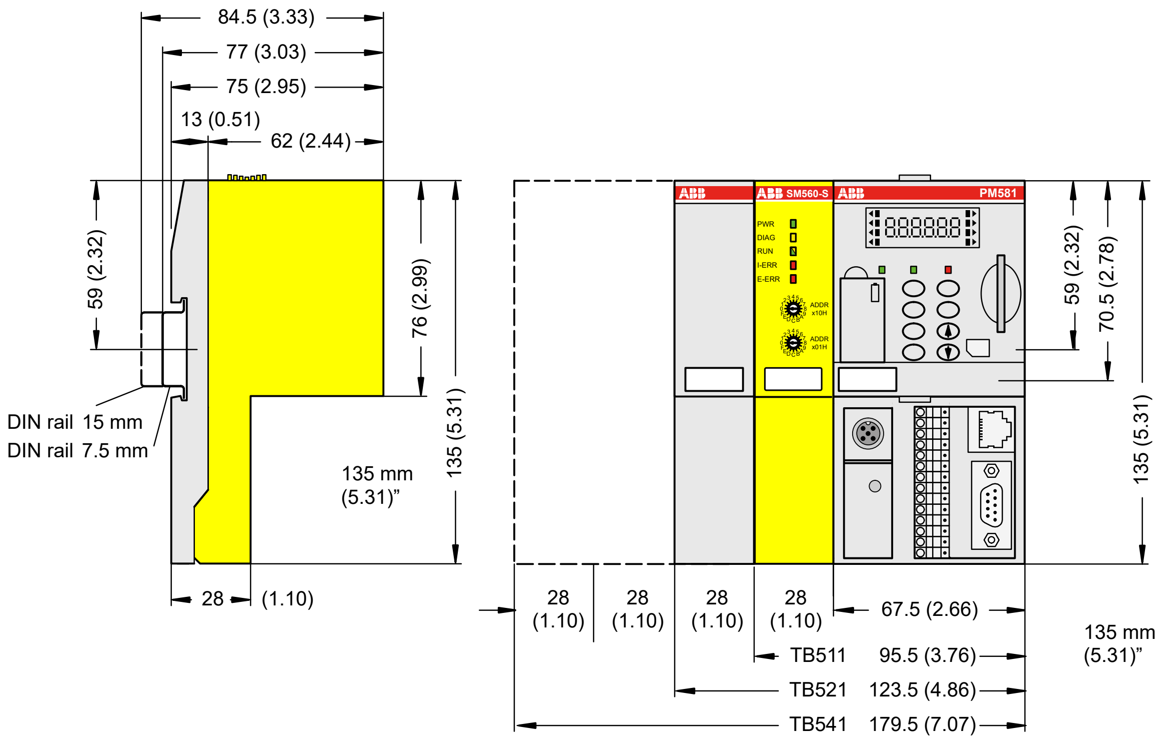


Fig. 9: Dimensions of the safety CPU

3.1.4 Diagnosis and LED status display

Safety CPU status is shown by its LEDs. RUN LED is bicolored. The following figure and table show positions and functions of 5 LEDs.

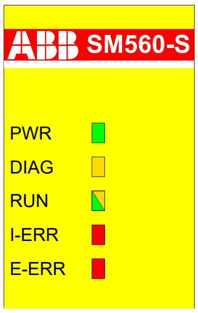


Fig. 10: LEDs for status display

Table 3: Status display and its meaning

LED	Description	Color	Status	Meaning
PWR	Module power supply	Green	ON	+3.3V internal power supply is available
			BLINKING	Not applicable
			OFF	+3.3V internal power supply is not available
DIAG	Diagnostics	Yellow	ON	Configuration error
			BLINKING	Not applicable
			OFF	No configuration error
RUN	Run mode indicator	Green	ON	Safety CPU is in RUN (safety) mode. The application program is executed.
			BLINKING	Not applicable
			OFF	Safety CPU is in DEBUG STOP (non-safety) mode. The application program is not executed.
		Yellow	ON	Safety CPU is in DEBUG RUN (non-safety) mode. The application program is executed.
			BLINKING	Firmware, boot project or boot code update indication
			OFF	Safety CPU is in DEBUG STOP (non-safety) mode. The application program is not executed.
I-ERR	Internal device error indicator	Red	ON	Internal device error leading to a SAFE STOP state (no valid PROFIsafe telegrams are generated by the device).
			BLINKING	Firmware or boot code update
			OFF	No internal device error leading to a safe state
E-ERR	External error indicator	Red	ON	This LED can be set only from the user application program using a special library POU SF_E_ERR_LED_SET ➮ *Chapter 4.6.7.1 "SF_E_ERR_LED_SET" on page 307*. One of possible use cases is the visualization of important external device errors.
			BLINKING	This LED can be set only from the user application program using a special library POU SF_E_ERR_LED_SET ➮ *Chapter 4.6.7.1 "SF_E_ERR_LED_SET" on page 307*. One of possible use cases is the visualization of light external device errors.
			OFF	No external errors were identified.

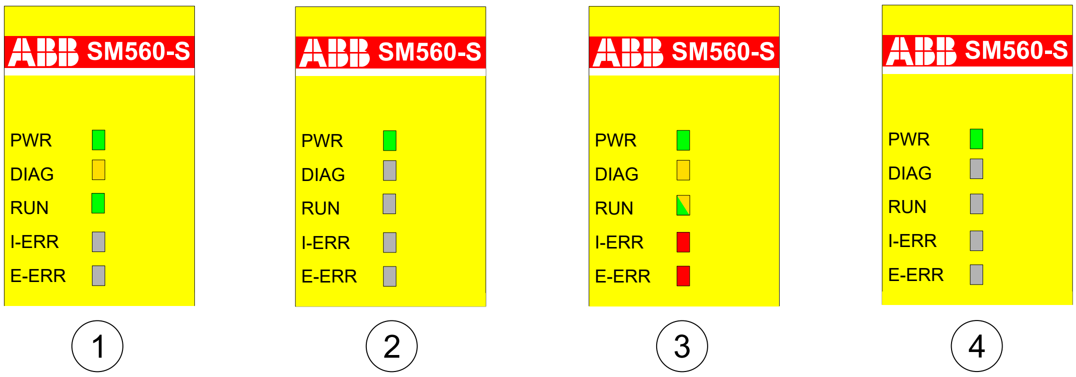


Fig. 11: LED states of the safety CPU during start-up

1 State 1 - Hardware reset
2 State 2 - Initialization
3 State 3 - LED test
4 State 4 - End of start-up

Error messages

Safety CPU error messages are aggregated together with other communication module error messages in non-safety CPUs. All error messages can be observed on non-safety CPU. In addition, error messages of the safety CPU can be observed on the safety CPU itself.

With AC500 V2 non-safety CPU: ➪ *Appendix B.2.1 "Error messages for safety CPUs" on page 375*

With AC500 V3 non-safety CPU: ➪ *Appendix C.2.1 "Error messages for safety CPUs" on page 392*

The complete list of AC500 error messages can be found in ➪ *[4]*.

> **NOTICE!**
> The error messages of not only the safety CPU but also of safety I/O modules are visualized on non-safety CPU display.
>
> No error message overflow on the safety CPU is possible. The maximum number of entries in the safety CPU diagnosis system is 100. If all 100 positions in the diagnosis system are occupied, the newest entries cause the oldest ones to be deleted from the diagnosis system on the safety CPU.
>
> After a power cycle of the safety CPU, error messages are deleted from the safety CPU diagnosis system.

3.1.5 Safety CPU module states

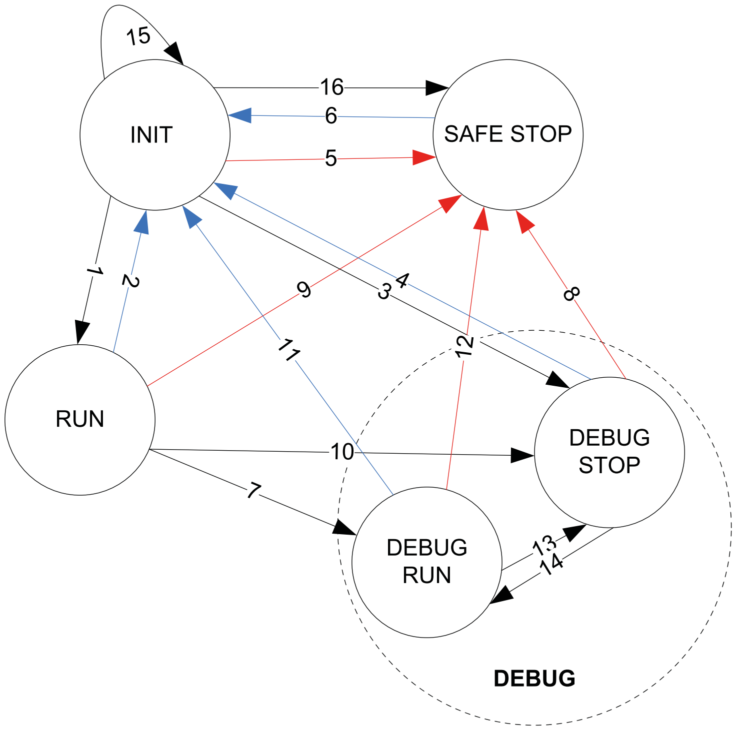


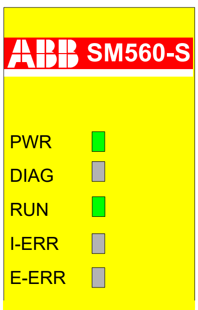
Fig. 12: Safety CPU states ➪ Chapter 3.1.5.1 "Description of safety CPU module states" on page 45 and transitions ➪ Chapter 3.1.5.2 "Transitions between safety CPU states " on page 47

Power cycle or *"reboot"* PLC browser/shell command on non-safety CPU
Errors of severity level 1 or 2
Further transitions

3.1.5.1 Description of safety CPU module states

INIT This is a temporary system state which is left after internal safety diagnostic tests and start-up procedures are executed. Refer to Fig. 11 on page 44 to see the LED states.

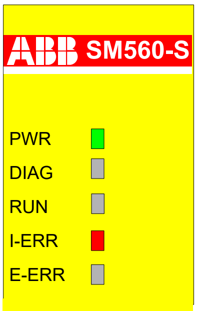
RUN



In this state, the safety application is normally executed, provided that the boot project is loaded. No error of severity levels 1 or 2 is available.

All CODESYS online services from "Online" menu are available for users, but only three of them can be executed without leaving RUN state: "Login", "Logout" and "Check boot project in PLC". All other services (e.g., set a breakpoint) switch the safety CPU to non-safety DEBUG states (DEBUG RUN or DEBUG STOP).

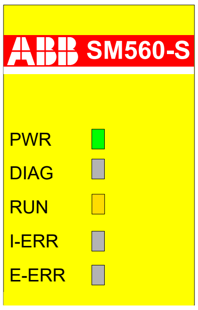
SAFE STOP



The safety CPU goes to SAFE STOP state if an error of severity level 1 or 2 is identified. All PROFIsafe output telegrams are nulled (no valid PROFIsafe telegrams are generated in this state). No CODESYS online services from "Online" menu are available for users.

This state can be left only after a power cycle or using *"reboot"* PLC browser/shell command on non-safety CPU.

DEBUG RUN



DEBUG RUN (non-safety) state can be reached if CODESYS online services from "Online" menu are used (except *"Login"*, *"Logout"* and *"Check boot project in PLC"*) from safe RUN state. The user can set a breakpoint in the safety program, perform *"Single cycle"* program execution, force and write variable values and execute other debugging functions of CODESYS Safety.

If CODESYS online service *"Stop"* is called or the breakpoint is reached in the safety application program, the safety CPU switches to DEBUG STOP (non-safety) state.

Valid PROFIsafe safety telegrams are generated in DEBUG RUN state. DEBUG RUN state is non-safe, thus, the responsibility for safe process operation lies entirely with the organization and person responsible for the activation of DEBUG RUN (non-safety) mode.

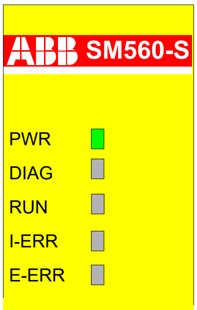
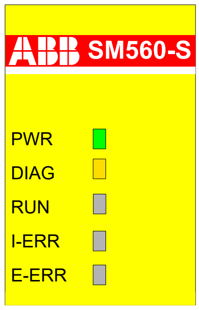
One can go back to a safe RUN state only after a power cycle or using *"reboot"* PLC browser/shell command on non-safety CPU.

DANGER!
The safety functionality and, as a result, safe process operation, is no more guaranteed by the safety CPU in the DEBUG RUN (non-safety) or DEBUG STOP (non-safety) mode.

In case of DEBUG RUN (non-safety) or DEBUG STOP (non-safety) mode activation on the safety CPU, **the responsibility for safe process operation lies entirely with the organization and person** responsible for the activation of DEBUG RUN (non-safety) or DEBUG STOP (non-safety) mode.

With the help of POU SF_SAFETY_MODE one can retrieve the information if the safety CPU is in SAFETY or DEBUG (non-safety) mode and, if required, stop or limit user application program execution *Chapter 4.6.7.7 "SF_SAFETY_MODE" on page 312*.

DEBUG STOP

Without error of severity level 3 or 4	With error of severity level 3 or 4
	

In this non-safe state, a user is able to intervene in safety program execution by setting breakpoints, etc., similar to DEBUG RUN state. The safety application program is not executed in DEBUG STOP (non-safety) state. The PROFIsafe F-Host and F-Devices (SM560-S-FD-1 and SM560-S-FD-4) of the safety CPU send PROFIsafe telegrams with fail-safe "0" values and set FV_activated for all safety I/O modules and F-Devices.

DANGER!
Since PROFIsafe F-Host continues to run in DEBUG STOP (non-safety) state, it is possible to reintegrate passivated safety I/O modules and bring them in the safety RUN state. One can force variables for safety I/O modules, for example, to activate safety outputs.

In case of or DEBUG RUN (non-safety) or DEBUG STOP (non-safety) mode activation on safety CPU, **the responsibility for safe process operation lies entirely with the organization and person** responsible for the activation of DEBUG RUN (non-safety) or DEBUG STOP (non-safety) mode.

If CODESYS online service *"RUN"* is called in the safety application program, the safety CPU switches to DEBUG RUN state.

All CODESYS online services are available in this state.

In case of CODESYS online commands *"Step in"*, *"Step over"*, *"Single cycle"* and when the breakpoint is reached, there is a switch between DEBUG RUN and DEBUG STOP states (transitions 13 and 14 in Fig. 12 on page 45).

One can go back to a safe RUN state only after power cycle or using *"reboot"* PLC browser/ shell command on non-safety CPU.

3.1.5.2 Transitions between safety CPU states

Transition (Fig. 12 on page 45)	**From**	**To**	**Description**
(1)	INIT	RUN	● Initialization was successful. ● Boot project is available and there is no configuration error or any other error of severity level 1 or 2.
(2)	RUN	INIT	Power cycle or *"reboot"* PLC browser/shell command from non-safety CPU.
(3)	INIT	DEBUG STOP	● Initialization was successful. ● No boot project is available or error of severity level 3. ● Switch address 0xFF was set on the safety CPU.
(4)	DEBUG STOP	INIT	Power cycle or *"reboot"* PLC browser/shell command from non-safety CPU.
(5)	INIT	SAFE STOP	● An error of severity level 1 or 2 was identified during the initialization. ● Unsuccessful firmware or boot code update.
(6)	SAFE STOP	INIT	Power cycle or *"reboot"* PLC browser/shell command from non-safety CPU.
(7)	RUN	DEBUG RUN	CODESYS Safety online service *"Toggle breakpoint"*, *"Write values"*, *"Force values"* or *"Single cycle"* was used.
(8)	DEBUG STOP	SAFE STOP	An error of severity level 1 or 2 was identified.
(9)	RUN	SAFE STOP	An error of severity level 1 or 2 was identified.

Transition (Fig. 12 on page 45)	From	To	Description
(10)	RUN	DEBUG STOP	● CODESYS Safety online services *“Stop”*, *“Sourcecode download”* or *“Reset”* (various). ● *[Run]* button on non-safety CPU was pressed (non-safety CPU was in "Run" state) . ● CODESYS online services *“Stop”* or *“Reset”* (various) on non-safety CPU. ● New safety boot project is loaded.
(11)	DEBUG RUN	INIT	Power cycle or *“reboot”* PLC browser/shell command from non-safety CPU.
(12)	DEBUG RUN	SAFE STOP	An error of severity level 1 or 2 was identified.
(13)	DEBUG RUN	DEBUG STOP	● CODESYS Safety online services *“Stop”* or *“Reset”* (various). ● *[Run]* button on non-safety CPU was pressed (non-safety CPU was in "Run" state). ● CODESYS online services *“Stop”* or *“Reset”* (various) on non-safety CPU. ● Breakpoint was reached during debugging. ● At the end of the safety CPU cycle in "Single cycle" debugging mode. ● New safety boot project is loaded.
(14)	DEBUG STOP	DEBUG RUN	● CODESYS Safety online services *“Step over”*, *“Step in”* and *“Run”*. ● CODESYS online service *“Run”*. ● *[Run]* button on non-safety CPU was pressed (non-safety CPU was in "Stop" state).
(15)	INIT	INIT	Power cycle or *“reboot”* PLC browser/shell command from non-safety CPU.
(16)	INIT	SAFE STOP	Switch address 0xFE, 0xFD or 0xFC was set on the safety CPU.

3.1.6 Safety and non-safety CPU interaction

The safety CPU and non-safety CPU have their own firmware, boot project and application program, which are executed separately. The only control element on non-safety CPU hardware, which allows changing the status of both non-safety and safety CPU is *[Run]* button on non-safety CPU. *[Run]* button on non-safety CPU can simultaneously stop and start both non-safety and safety CPU. This behavior of *[Run]* button depends on non-safety CPU settings ⮫ *[4]*. Stopped safety CPU means that application program execution has stopped only. PROFIsafe F-Host and F-Device stacks ⮫ *[3]* continues to run in fail-safe mode. All safety I/O modules are passivated and substitute values "0" are used for safety I/Os and F-Devices. PROFIsafe F-Host and F-Device stack execution can be stopped by entering SAFE STOP state only. In this case, PROFIsafe telegrams are not generated and I-ERR LED is on.

DANGER!

It is not possible to safely start safety CPU using *[Run]* button on non-safety CPU. The safety CPU always goes to non-safe DEBUG mode (DEBUG RUN or DEBUG STOP) as soon as *[Run]* button is pressed on non-safety CPU *Chapter 3.1.5.1 "Description of safety CPU module states" on page 45*. To bring the safety CPU back into the safe RUN mode, perform a power cyle of the safety CPU or use *"reboot"* PLC browser/shell command on non-safety CPU.

The commands *"Run"* and *"Stop"* in engineering suite have the same effect on the safety CPU and non-safety CPU as *[Run]* button on non-safety CPU.

There are some parameters of non-safety CPU configuration which influence the overall system behavior of safety and non-safety CPU *Appendix B.3 "AC500 V2 non-safety CPU parameters configuration" on page 382* *Appendix C.3 "AC500 V3 non-safety CPU parameters configuration" on page 397*.

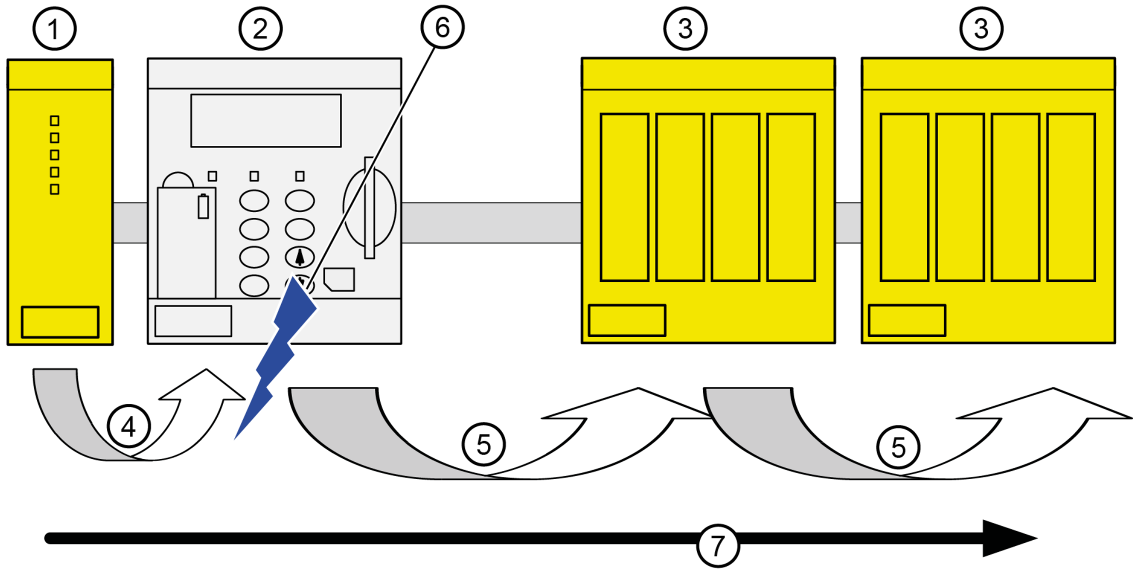


Fig. 13: Influence of non-safety CPU parameter settings on safety telegram flow

1 Safety CPU
2 Non-safety CPU
3 Safety I/O module
4 Valid safety telegram
5 Telegram with "0" values or valid safety telegram
6 Non-safety CPU settings
7 Safety CPU safety telegrams with output values

3.1.7 Technical data

Additional technical data is available in ABB PLC catalog at www.abb.com/plc.

NOTICE!

Safety CPU -XC version is available for usage in extreme environmental conditions *Appendix A "System data for AC500-S-XC" on page 367*.

Memory

Data	Value	Unit
User program memory of SM560-S	1	MB

Data	Value	Unit
User program memory of SM560-S-FD-1 and SM560-S-FD-4	1.3	MB
User data memory (thereof 120 kB saved)	1	MB

Performance

Data	Value	Unit
Cycle time - binary	0.05	µs/instruction
Cycle time - word	0.06	µs/instruction
Cycle time - floating-point	0.50	µs/instruction

Voltages, according to EN 61131-2

Data	Value	Unit
Process and supply voltage (without ripple)	24 (-15 %, +20 %)	V DC
Absolute limits (including ripple)	19.2 ... 30	V DC
Ripple	< 5	%
Protection against reverse polarity	10	s

DANGER!
Exceeding the permitted process or supply voltage range (< -35 V DC or > +35 V DC) could lead to unrecoverable damage of the system.

Allowed interruptions of power supply, according to EN 61131-2

Data	Value	Unit
DC supply interruptions	< 10	ms
Time between 2 DC supply interruptions, PS2	> 1	s

Environmental conditions

Data	Value	Unit
Operating temperature*	0 ... +60	°C
Storage temperature	-40 ... +85	°C
Transport temperature	-40 ... +85	°C
Humidity without condensation	max. 95	%
Operating air pressure	> 800	hPa
Storage air pressure	> 660	hPa
Operating altitude	< 2000	m above sea level
Storage altitude	< 3500	m above sea level

* Extended temperature ranges (below 0 °C and above +60 °C) can be supported in special versions of the safety CPU *Appendix A “System data for AC500-S-XC” on page 367*.

Creepage distances and clearances The creepage distances and clearances meet the overvoltage category II, pollution degree 2.

Power supply units For the supply of modules, power supply units according to PELV/SELV specifications must be used.

Electromagnetic compatibility For information on electromagnetic compatibility refer to the latest TÜV SÜD Report ↬ *[2]*.

Mechanical properties

Data	Value	Unit
Mounting	horizontal (or vertical with derating (maximal operating temperature reduced to +40 °C))	
Degree of protection	IP 20	
Housing	according to UL 94	
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 3.5 mm	2 ... 15	Hz
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 1 g *	15 ... 150	Hz
Shock test (all three axes), 11 ms half-sinusoidal	15	g
MTBF	168	years

* Higher values on request

Self-test and diagnostic functions Start-up and runtime tests: Program flow control, RAM, CPU, etc.

Dimensions, weight

Data	Value	Unit
W x H x D	28 x 135 x 75	mm
Weight	~ 100	g

Certifications CE, cUL (further certifications at *www.abb.com/plc*)

3.1.8 Ordering data

Type	Description	Part no.
SM560-S	Safety module - CPU, safety related module up to SIL 3	1SAP 280 000 R0001
SM560-S-XC	Safety module - CPU, safety related module up to SIL 3, extreme conditions	1SAP 380 000 R0001
SM560-S-FD-1	Safety module - CPU, safety related module up to SIL 3 with F-Device functionality for 1 PROFIsafe network	1SAP 286 000 R0001

Type	Description	Part no.
SM560-S-FD-1-XC	Safety module - CPU, safety related module up to SIL 3 with F-Device functionality for 1 PROFIsafe network, extreme conditions	1SAP 386 000 R0001
SM560-S-FD-4	Safety module - CPU, safety related module up to SIL 3 with F-Device functionality for up to 4 PROFIsafe networks	1SAP 286 100 R0001
SM560-S-FD-4-XC	Safety module - CPU, safety related module up to SIL 3 with F-Device functionality for up to 4 PROFIsafe networks, extreme conditions	1SAP 386 100 R0001

3.2 Generic safety I/O module behavior

3.2.1 Overview

All safety I/O modules (AI581-S, DI581-S and DX581-S) can be used in a centralized or remote configuration with PROFINET/PROFIsafe (Fig. 3 on page 21). PROFINET devices CI501-PNIO, CI502-PNIO, CI504-PNIO and CI506-PNIO can be used to attach safety I/O modules in remote configurations. Safety I/O modules can be freely mixed with any non-safety I/Os from AC500 and AC500-eCo product families.

NOTICE!
Safety I/O module firmware update can be currently performed only by the qualified personnel in the ABB factory.

3.2.2 Safety I/O module states

Safety I/O module system states can be described using the following two state charts.

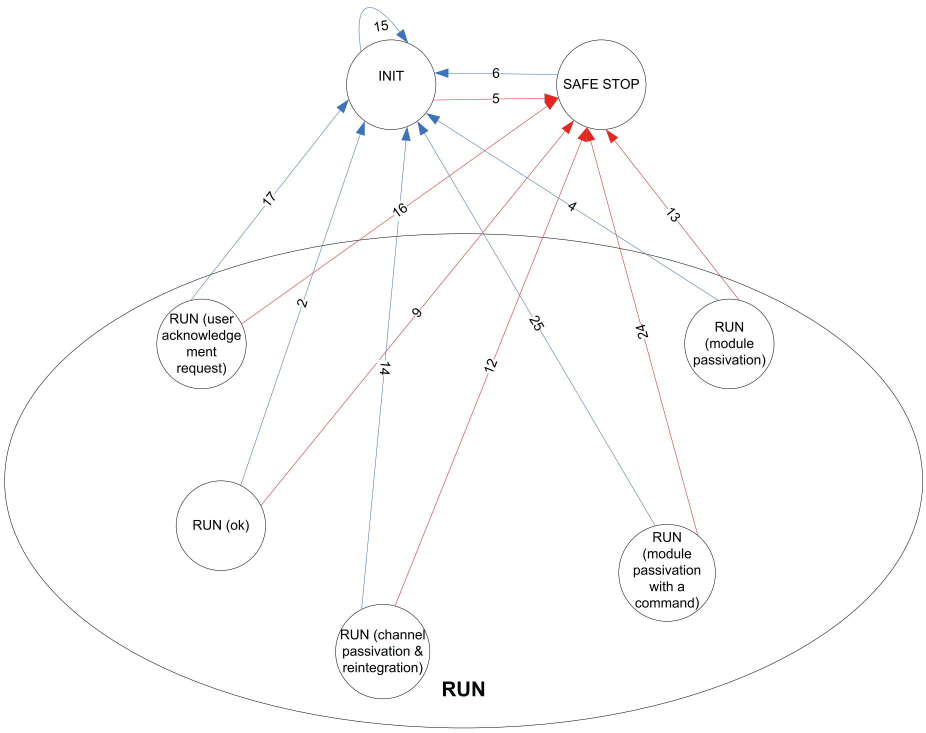


Fig. 14: Overview of transitions related to power cycles and errors of severity level 1 in safety I/O modules

Power cycle
Error of severity level 1

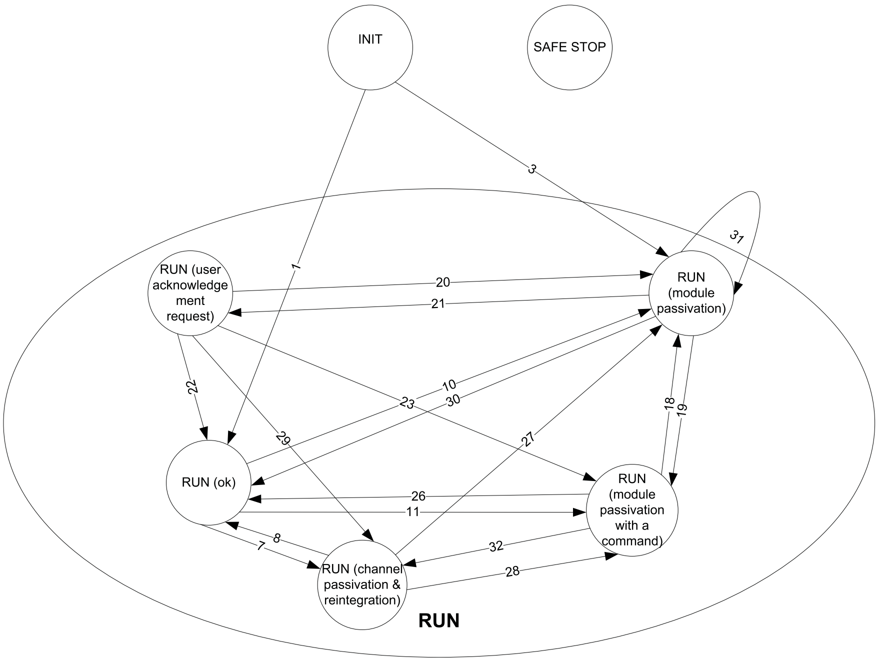


Fig. 15: Overview of transitions in safety I/O modules (except power cycles and errors of severity level 1)

Transitions

3.2.2.1 Description of safety I/O module states

INIT The hardware is initialized and internal start-up tests of the safety I/O module are executed. Refer to Fig. 16 on page 62 to see the LED states. After a successful parameterization, the PROFIsafe communication is expected to be initiated by the PROFIsafe F-Host.

The safety I/O module will remain in this state:

- as long as the undervoltage is detected.
- if the parameterization failed or pending.
- if the PROFIsafe communication is pending.

Users have to check that a dedicated qualifier output bit (PROFIsafe diagnostic) for at least one of the channels in the given safety I/O module is set to "1" to verify that PROFIsafe F-Devices are initialized.

PROFIsafe status bits in the F-Host for safety I/O module:

OA_Req_S = 0

FV_activated_S = 1

Device_Fault = 0

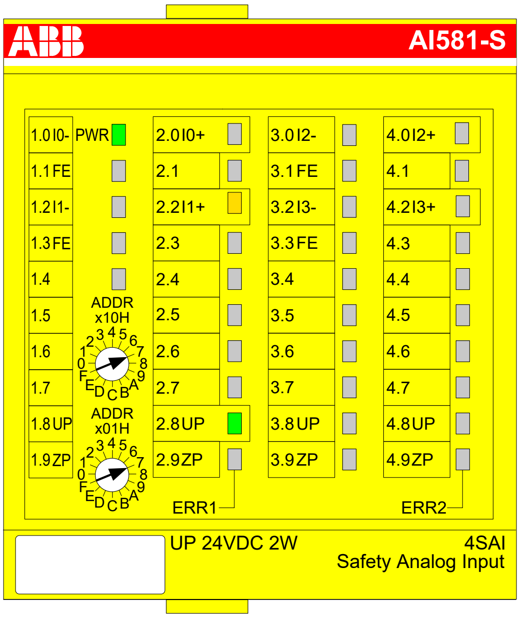
Process data bits in the safety I/O module process image:

PROFIsafe diagnostic bit = 0

Channel process value = 0

Reintegration request bit = 0

RUN (ok)



PROFIsafe communication is up and running. The safety application is running without any detected errors.

PROFIsafe status bits in the F-Host for safety I/O module:

OA_Req_S = 0

FV_activated_S = 0

Device_Fault = 0

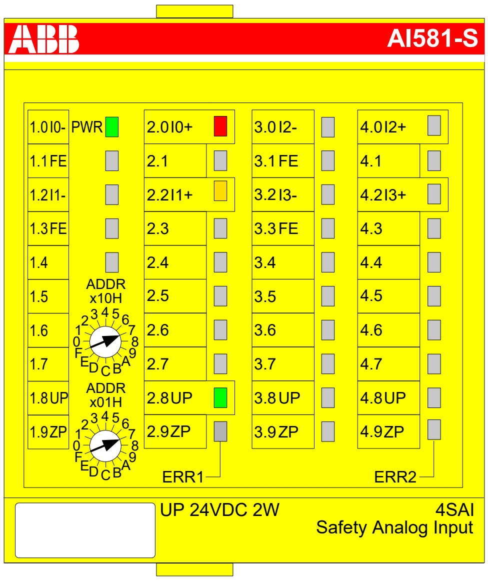
Process data bits in the safety I/O module process image:

PROFIsafe diagnostic bit = 1

Channel process value = Process value

Reintegration request bit = 0

RUN (channel passivation and reintegration)



PROFIsafe communication is up and running. The safe application is running with detected channel errors.

Channel error (e.g., no expected test pulses, discrepancy time, etc.) is identified in at least one of channels. The fail-safe value ("0") is transferred to the PROFIsafe F-Host for the passivated input channel(s). The related PROFIsafe diagnostic bit(s) are also set to "0" to indicate the usage of fail-safe values.

A passivated output channel has a state of "0" and the related PROFIsafe diagnostic bit(s) are also set to "0" to indicate the usage of fail-safe values.

As soon as the channel error is gone (e.g., wiring error was corrected; this is valid only for those errors which are acknowledgeable), the reintegration request bit for the given channel switches to "1", which indicates the safety application running on the safety CPU that a reintegration of the channel is possible. Setting the acknowledge reintegration bit from "0" to "1" initiates a reintegration of the given channel. A positive edge from "0" to "1" is required to acknowledge channel reintegration.

As soon as all channel errors are gone and acknowledged, the RUN (ok) state is reached.

PROFIsafe status bits in the F-Host for safety I/O module:

OA_Req_S = 0

FV_activated_S = 0

Device_Fault = 0

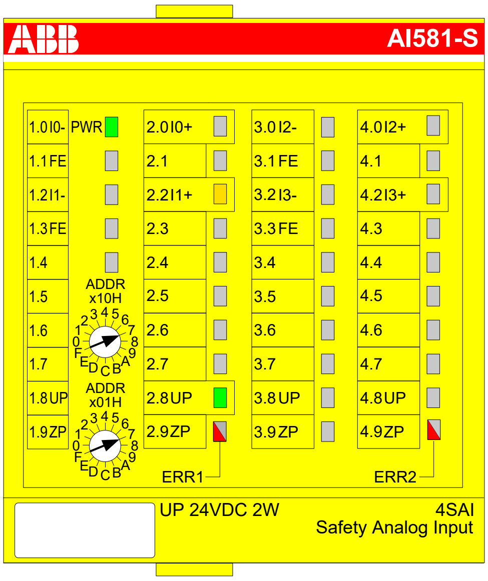
Process data bits in the safety I/O module process image:

PROFIsafe diagnostic bit = 0

Channel process value = 0

Reintegration request bit = 0 if an error is still present; 1 if the channel can be reintegrated.

RUN (module passivation): alternating blinking of ERR1 and ERR2 LEDs



PROFIsafe communication is up and running. The safety application is running with a present module error.

The module and, as a result, all its channels are passivated. Possible reasons for module passivation are:

- PROFIsafe communication failure (CRC error)
- PROFIsafe watchdog timeout exceeded
- Undervoltage/overvoltage detected (Device_Fault status bit = 1)

The fail-safe value "0" is transferred to the safety PLC for all passivated input channels, if the connection to the PROFIsafe F-Host is possible. The safety application continuously attempts to establish a communication to the safety CPU, if the communication is broken. All passivated output channels have a state of "0".

A state transition to another RUN mode is only possible if the detected error is gone.

PROFIsafe status bits in the F-Host for safety I/O module (if communication is possible!):

OA_Req_S = 0

FV_activated_S = 1

Device_Fault = 1 (in case of undervoltage/overvoltage detected) and/or CE_CRC = 1 (in case of communication error) and/or WD_timeout = 1 (in case of watchdog timeout)

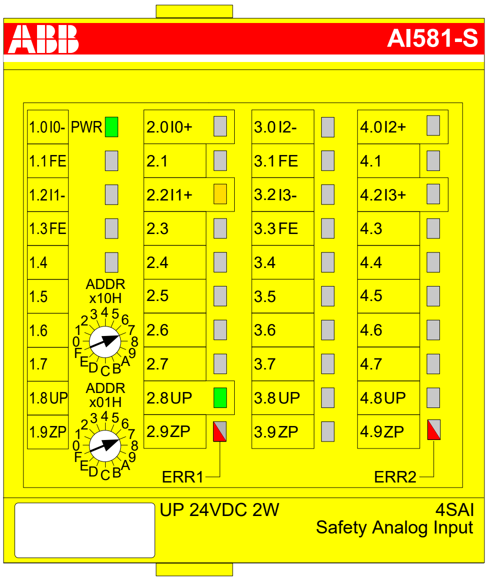
Process data bits in the safety I/O module process image:

PROFIsafe diagnostic bit = 0

Channel process value = 0

Reintegration request bit = 0

RUN (module passivation with a command): alternating blinking of ERR1 & ERR2 LEDs



PROFIsafe communication is up and running. The safety application is running without any detected errors.

The module and all its channels are passivated because the safety application on the safety CPU requested a module passivation (activate_FV_C = 1 was set).

The fail-safe value "0" is transferred to the safety CPU for all passivated input channels. All passivated output channels have a state of "0". The PROFIsafe diagnostic bit(s) for all channels have the state of "0" to indicate that fail-safe values are transferred.

PROFIsafe status bits in the F-Host for safety I/O module:

FV_activated_S = 1

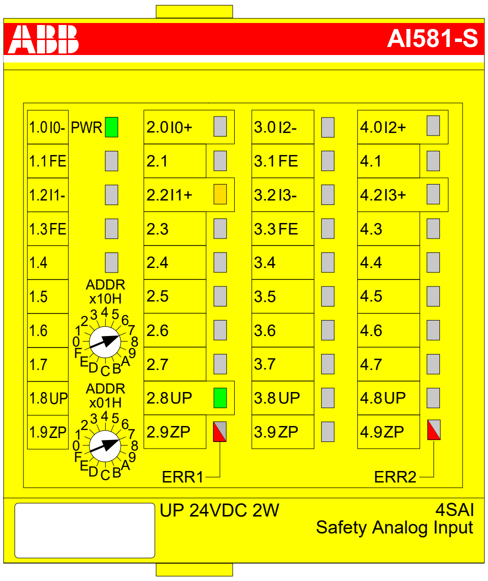
Process data bits in the safety I/O module process image:

PROFIsafe diagnostic bit = 0

Channel process value = 0

Reintegration request bit = 0

RUN (user acknowledgment request): alternating blinking of ERR1 & ERR2 LEDs



PROFIsafe communication is up and running. The safety application is running without any errors but waits for the acknowledgment of a module reintegration (module error is gone).

The fail-safe value "0" is still transferred to the safety CPU for all passivated input channels. All passivated output channels have a state of "0". The PROFIsafe diagnostic bits for all channels have the state of "0" to indicate that fail-safe values are transferred.

The OA_Req_S bit is reported as "1".

As soon as the safety application of the safety CPU sets OA_C (positive edge), the safety I/O module goes to RUN (ok) state if no further errors are detected. One has to send the positive edge to the safety I/O module until OA_Req_S starts delivering "0".

PROFIsafe status bits in the F-Host for safety I/O module:

OA_Req_S = 1

FV_activated_S = 1

Device_Fault = 0

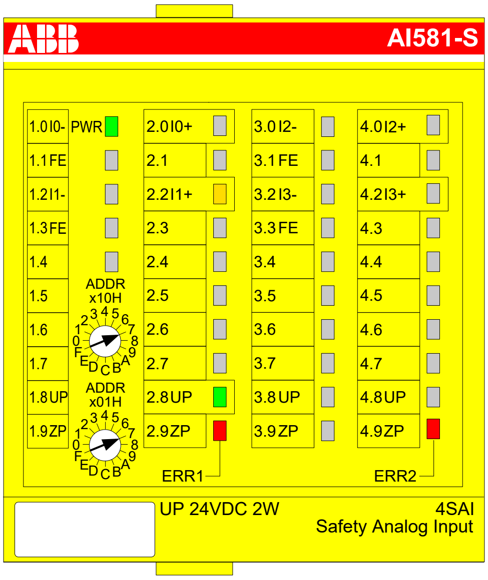
Process data bits in the safety I/O module process image:

PROFIsafe diagnostic bit = 0

Channel process value = 0

Reintegration request bit = 0

SAFE STOP



The safety application execution was stopped. No PROFIsafe communication is possible.

This state is reached if an error of severity level 1 (e.g., CPU test, RAM test, etc. failed) took place.

This state can be left only through a power cycle or *"reboot"* command from non-safety CPU or communication interface module.

PROFIsafe status bits in the F-Host for safety I/O module:

OA_Req_S = 0

FV_activated_S = 1

Device_Fault = 0

Process data bits in the safety I/O module process image:

PROFIsafe diagnostic bit = 0

Channel process value = 0

Reintegration request bit = 0

3.2.2.2 Transitions between safety I/O module states

Transition (Fig. 14 on page 53, Fig. 15 on page 53)	From	To	Description
(1)	INIT	RUN (ok)	Safety I/O module comes to this state directly after INIT during a normal start-up
(2)	RUN (ok)	INIT	Power cycle
(3)	INIT	RUN (module passivation)	PROFIsafe watchdog, PROFIsafe communication error or undervoltage/overvoltage was detected directly after INIT. The safety I/O module can reach this state also after a power cycle of the safety I/O module if safety CPU with PROFIsafe F-Host continues running and brings safety I/O module to a fail-safe RUN (module passivation) state after a power cycle.
(4)	RUN (module passivation)	INIT	Power cycle
(5)	INIT	SAFE STOP	Error(s) of severity level 1 (CPU test, RAM test, etc. failed) took place
(6)	SAFE STOP	INIT	Power cycle
(7)	RUN (ok)	RUN (channel passivation and reintegration)	Channel error was identified by the safety I/O module. The tests (whenever it is possible) are continued for the given channel to be able to see if the error is gone (e.g., wiring error was corrected). As soon as the error is gone, the module sets "Reintegration request" bit = "1" for the given channel.
(8)	RUN (channel passivation and reintegration)	RUN (ok)	● The channel error is gone. ● "Reintegration request" bit = 1 is set for the given channel by the safety I/O module. ● "Acknowledge reintegration" bit (positive edge) is set by PROFIsafe F-Host for the given channel.
(9)	RUN (ok)	SAFE STOP	Error(s) of severity level 1 (CPU test, RAM test, etc. failed) took place
(10)	RUN (ok)	RUN (module passivation)	PROFIsafe watchdog, PROFIsafe communication error or undervoltage/overvoltage was detected.
(11)	RUN (ok)	RUN (module passivation with a command)	"activate_FV_C = 1" command was sent from the safety CPU
(12)	RUN (channel passivation and reintegration)	SAFE STOP	Error(s) of severity level 1 (CPU test, RAM test, etc. failed) took place
(13)	RUN (module passivation)	SAFE STOP	Error(s) of severity level 1 (CPU test, RAM test, etc. failed) took place
(14)	RUN (channel passivation and reintegration)	INIT	Power cycle
(15)	INIT	INIT	Power cycle
(16)	RUN (user acknowledgment request)	SAFE STOP	Error(s) of severity level 1 (CPU test, RAM test, etc. failed) took place

Transition (Fig. 14 on page 53, Fig. 15 on page 53)	From	To	Description
(17)	RUN (user acknowledgment request)	INIT	Power cycle
(18)	RUN (module passivation with a command)	RUN (module passivation)	PROFIsafe watchdog, PROFIsafe communication error or undervoltage/overvoltage was identified. **Note:** In this transition, it is possible that WD_timeout bit of PROFIsafe F-Host instance toggles if watchdog timeout is periodically recognized by the safety I/O module.
(19)	RUN (module passivation)	RUN (module passivation with a command)	If the threshold shut-down value was not reached during process undervoltage or overvoltage phase and the process voltage is back in the normal range, the safety I/O module reintegrates and would go to RUN (ok) state automatically, but short time before the "activate_FV_C = 1" command was sent from the PROFIsafe F-Host stack, which leads the safety I/O module to RUN (module passivation with a command) state.
(20)	RUN (user acknowledgment request)	RUN (module passivation)	Process undervoltage/overvoltage was identified.
(21)	RUN (module passivation)	RUN (user acknowledgment request)	● Module error (watchdog or communication error (CRC)) is gone. **and** ● Command activate_FV_C = 0 **then** ● Safety I/O module sets OA_Req_S = 1
(22)	RUN (user acknowledgment request)	RUN (ok)	● OA_Req_S = 1 was set by the safety I/O module after the module error is gone. ● OA_C (positive edge) was set by the PROFIsafe F-Host for the given safety I/O module.
(23)	RUN (user acknowledgment request)	RUN (module passivation with a command)	"activate_FV_C = 1" command was sent from the PROFIsafe F-Host
(24)	RUN (module passivation with a command)	SAFE STOP	Error(s) of severity level 1 (CPU test, RAM test, etc. failed) took place
(25)	RUN (module passivation with a command)	INIT	Power cycle
(26)	RUN (module passivation with a command)	RUN (ok)	● No module error ● Command activate_FV_C = 0
(27)	RUN (channel passivation and reintegration)	RUN (module passivation)	PROFIsafe watchdog, PROFIsafe communication error or undervoltage/overvoltage was detected. **Note:** In this transition, it is possible that WD_timeout bit of PROFIsafe F-Host instance toggles if watchdog timeout is periodically recognized by the safety I/O module.

Transition (Fig. 14 on page 53, Fig. 15 on page 53)	From	To	Description
(28)	RUN (channel passivation and reintegration)	RUN (module passivation with a command)	"activate_FV_C = 1" command was sent from the PROFIsafe F-Host stack
(29)	RUN (user acknowledgment request)	RUN (channel passivation and reintegration)	This transition is possible only if channel error was identified before or during module passivation. As a result, after module reintegration one of the channel tests directly brings safety I/O module to RUN (channel passivation and reintegration state).
(30)	RUN (module passivation)	RUN (ok)	If the threshold shut-down value was not reached during undervoltage phase and the process voltage is back in the normal range, the safety I/O module reintegrates and goes to RUN (ok) state automatically. If the threshold fuse value was not reached during overvoltage phase and the process voltage is back in the normal range, the safety I/O module reintegrates and goes to RUN (ok) state automatically.
(31)	RUN (module passivation)	RUN (module passivation)	If process undervoltage event was detected two times within 1 s, then the safety I/O module remains in RUN (module passivation) state.
(32)	RUN (module passivation with a command)	RUN (channel passivation and reintegration)	This transition is possible only if channel error was identified during RUN (module passivation with a command) state. As a result, after command activate_FV_C = 0, safety I/O module goes to RUN (channel passivation and reintegration state).

3.2.3 Undervoltage / overvoltage

If undervoltage (< 18 V) is detected in the safety I/O module, the module goes to RUN (module passivation) state, until the process voltage did not reach the threshold shut-down value (16 V), when no further communication to PROFIsafe F-Host is possible. If the threshold shut-down value (16 V) was not reached during undervoltage phase and the process voltage is back in the normal range (≥ ~18 V), the safety I/O module reintegrates and goes to RUN (ok) state automatically.

To avoid unintended permanent module passivation and reintegration, the following feature is available for undervoltage case:

- The user has to continuously supervise Device_Fault bit of the safety I/O module and if Device_Fault = 1 is detected, he passivates the module with activate_FV_C = 1.

If overvoltage (> 31.2 V) is detected in the safety I/O module, the module goes to RUN (module passivation) state, until the process voltage did not reach the threshold fuse value (> 35 V) when the safety I/O module is damaged and has to be replaced. If the threshold fuse value was not reached during overvoltage phase and the process voltage is back in the normal range, the safety I/O module reintegrates and goes to RUN (ok) state automatically. To avoid unintended permanent module passivation and reintegration, the same feature (supervision of Device_Fault bit) as for undervoltage is available.

3.2.4 Diagnosis

> **DANGER!**
> The diagnosis data is not safety-relevant and, thus, shall not be used in safety application program for execution of safety functions.

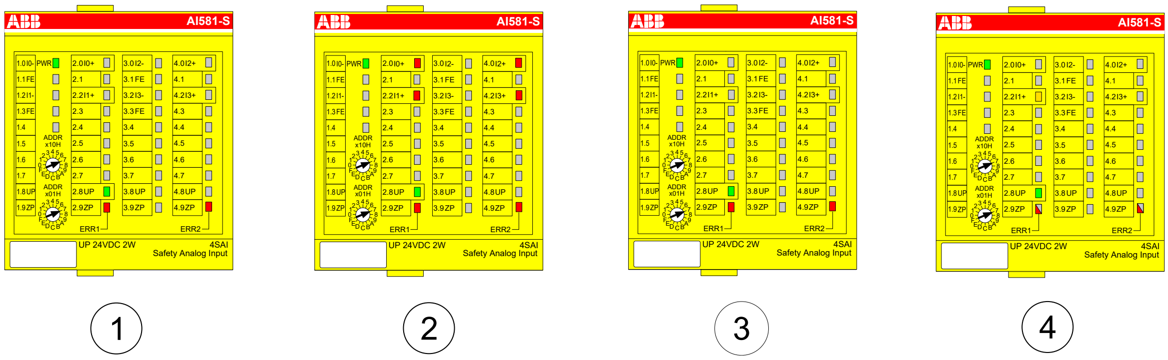


Fig. 16: LED states of safety I/O modules during start-up (example with AI581-S module)

1 State 1 - Hardware reset and initialization
2 State 2 - LED test
3 State 3 - End state of initialization
4 State 4 - Parameterization is complete, but no PROFIsafe communication yet

Error messages

> **NOTICE!**
> External errors (wiring or sensor errors) in safety I/O modules lead to the channel passivation ("0" values are delivered). As soon as an external error is fixed and this is recognized by internal safety I/O module tests, safety I/O module channels request an acknowledgment for their reintegration to the normal safety process control mode. The user can acknowledge such channels using dedicated channel bits , refer to Fig. 73 on page 146.

Safety I/O module error messages are aggregated together with other module error messages in non-safety CPU.

With AC500 V2 non-safety CPU: *Appendix B.2.2 “Error messages for safety I/O modules” on page 380*

With AC500 V3 non-safety CPU: *Appendix C.2.2 “Error messages for safety I/O modules” on page 395*

The complete list of AC500 error messages can be found in *[4]*.

3.3 DI581-S safety digital input module

Elements of the module

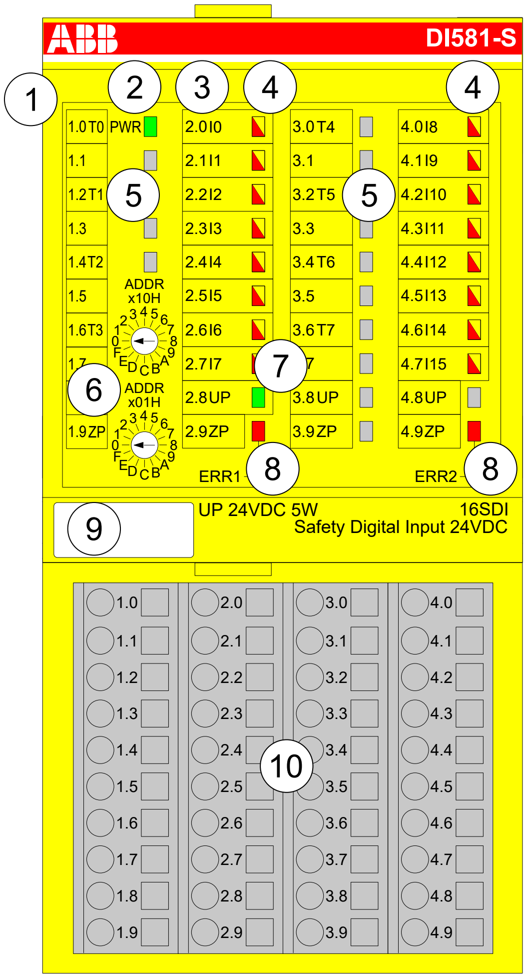


Fig. 17: Safety digital input module DI581-S, plugged on terminal unit TU582-S

1 I/O bus
2 System LED
3 Allocation terminal no. - signal name
4 16 yellow/red LEDs signal status I0 ... I7/I8 ... I15
5 8 unique phase-shifted test pulse outputs T0 ... T3/T4 ... T7
6 2 rotary switches for PROFIsafe address
7 Green LED for process voltage UP
8 Red LEDs to display module errors
9 Label (TA525)
10 I/O terminal unit (TU582-S)

3.3.1 Purpose

Safety digital input module DI581-S can be used as a remote expansion module at CI501-PNIO, CI502-PNIO, CI504-PNIO and CI506-PNIO PROFINET modules or locally at AC500 CPUs for up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1) safety applications.

> **NOTICE!**
> SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1) reachable in your safety application depend on the wiring of your sensors to DI581-S module *Chapter 3.3.7 “Circuit examples DI581-S” on page 71*.

DI581-S contains 16 safety digital inputs 24 V DC separated in two groups (2.0 ... 2.7 and 4.0 ... 4.7) with no potential separation between the channels.

The inputs are not electrically isolated from the other electronic circuitry of the module.

3.3.2 Functionality

Digital inputs	16 (24 V DC)
LED displays	for signal status, module errors, channel errors and supply voltage
Internal power supply	through the expansion bus interface (I/O bus)
External power supply	via the terminals ZP and UP (process voltage 24 V DC)

Self-tests and diagnostic functions (both start-up and runtime), like CPU and RAM tests, program flow control, cross-talk and stuck-at-1 tests, etc. are implemented in DI581-S according to IEC 61508 SIL 3 requirements.

> **NOTICE!**
> Only F_Dest_Add is used for PROFIsafe F-Device identification in DI581-S.

DI581-S contains 16 safety digital input channels with the following features:

- Phase-shifted (unique) test pulses T0 ... T7 can be used for connection of mechanical sensors. Test pulse outputs T0 ... T7 provide 24 V signal with a short phase-shifted unique pulses (0 V) of 1 ms. Since the test pulses on each of the test pulse output channels are unique (due to the phase shift), they can be used to monitor the cross-talk between the given input channel with connected test pulse output and another wire, e.g, with 24 V DC, another test pulse output, etc. Test pulse outputs are dedicated ones:
  – T0 can be used only with input channels I0 and I1
  – T1 can be used only with input channels I2 and I3
  – T2 can be used only with input channels I4 and I5
  – T3 can be used only with input channels I6 and I7
  – T4 can be used only with input channels I8 and I9
  – T5 can be used only with input channels I10 and I11
  – T6 can be used only with input channels I12 and I13
  – T7 can be used only with input channels I14 and I15
- Input delay with the following values: 1 ms, 2 ms, 5 ms, 10 ms, 15 ms, 30 ms, 50 ms, 100 ms, 200 ms, 500 ms. Input delay value of 1 ms is the minimum one.

> **NOTICE!**
> The allowed signal frequency on safety digital inputs is dependent on the input delay value for the given channel:
>
> – For channel input delay values of 1 ... 10 ms, the pulse length of input signal shall be ≥ 15 ms (~ 65 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 15 ms, the pulse length of input signal shall be ≥ 20 ms (~ 50 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 30 ms, the pulse length of input signal shall be ≥ 40 ms (~ 25 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 50 ms, the pulse length of input signal shall be ≥ 60 ms (~ 15 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 100 ms, the pulse length of input signal shall be ≥ 120 ms (~ 8 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 200 ms, the pulse length of input signal shall be ≥ 250 ms (~ 4 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 500 ms, the pulse length of input signal shall be ≥ 600 ms (~ 1.5 Hz) to avoid occasional input channel passivation.

DANGER!
The input delay parameter means that signals with the duration shorter than input delay value are always not captured by the safety module.

The signals with the duration of equal to or longer than "input delay parameter" + "input delay accuracy" are always captured by the safety module, provided that the allowed frequency (refer to previous notice) of the safety input signal is not exceeded.

The "input delay accuracy" can be estimated based on the following assumptions:

– If no test pulses are configured for the given safety digital input, then input delay accuracy can be calculated as 1 % of set input delay value (however, input delay accuracy value must be at least 0.5 ms!).
– If test pulses are configured for the given safety digital input of DI581-S module, then the input delay accuracy values can be estimated based on the input delay parameter value *Table 4 “Input delay accuracy for DI581-S” on page 65*.

Table 4: Input delay accuracy for DI581-S

Input delay (ms)	Input delay accuracy (ms)
1	2
2	2
5	3
10	4
15	5
30	6
50	7
100	10
200	15
500	25

- Checking of process power supply (diagnostic message is sent from the safety I/O module to the CPU informing about the lack of process power supply for the given safety I/O module). This function is a non-safety one and is not related to the internal safety-relevant over- and undervoltage detection.
- 2 channel equivalent and 2 channel antivalent mode with discrepancy time monitoring (configurable 10 ms ... 30 s).

NOTICE!
In a 2 channel mode, the lower channel (channels 0/8 ➔ Channel 0, channels 1/9 ➔ Channel 1, etc.) transports the aggregated process value, PROFIsafe diagnostic bit, acknowledgment request and acknowledge reintegration information. The higher channel always provides the passivated value "0".

DANGER!

After discrepancy time error, the relevant channels are passivated. As soon as a valid sensor state is observed (equivalent or antivalent, depending on the selected mode), reintegration request status bit for the given channel becomes TRUE. You can acknowledge an error using acknowledge reintegration command bit for the given channel. This can directly lead to the machine start, because both TRUE - TRUE and FALSE - FALSE are valid states for equivalence and TRUE - FALSE and FALSE - TRUE are valid states for antivalence.

Make sure that such behavior is acceptable in your safety application. If no, then you can use either included PLCopen Safety POUs for 2 channel evaluation in your safety program or write your own POUs for 2 channel evaluation on the safety CPU.

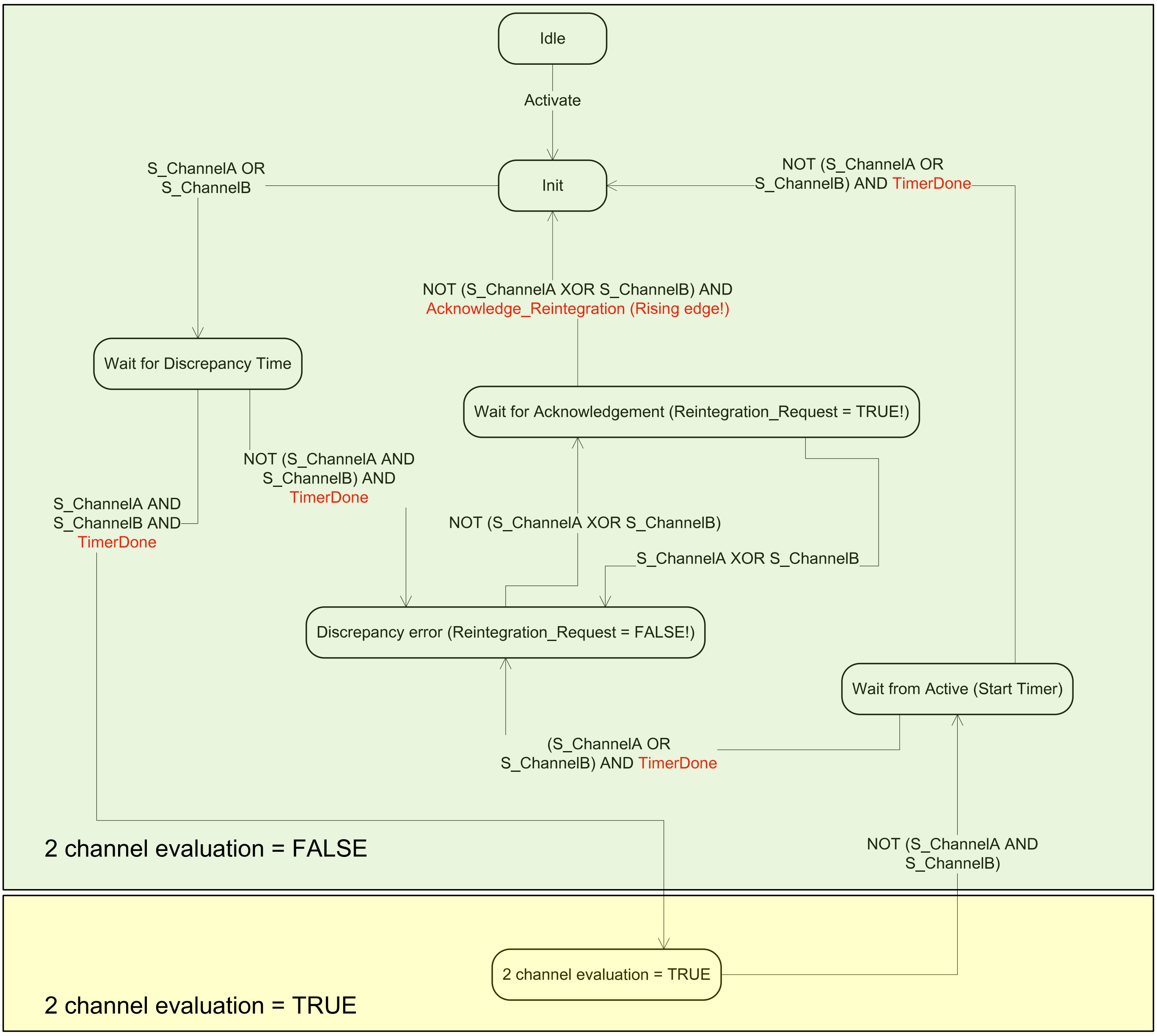


Fig. 18: 2 channel equivalent mode implemented in DI581-S

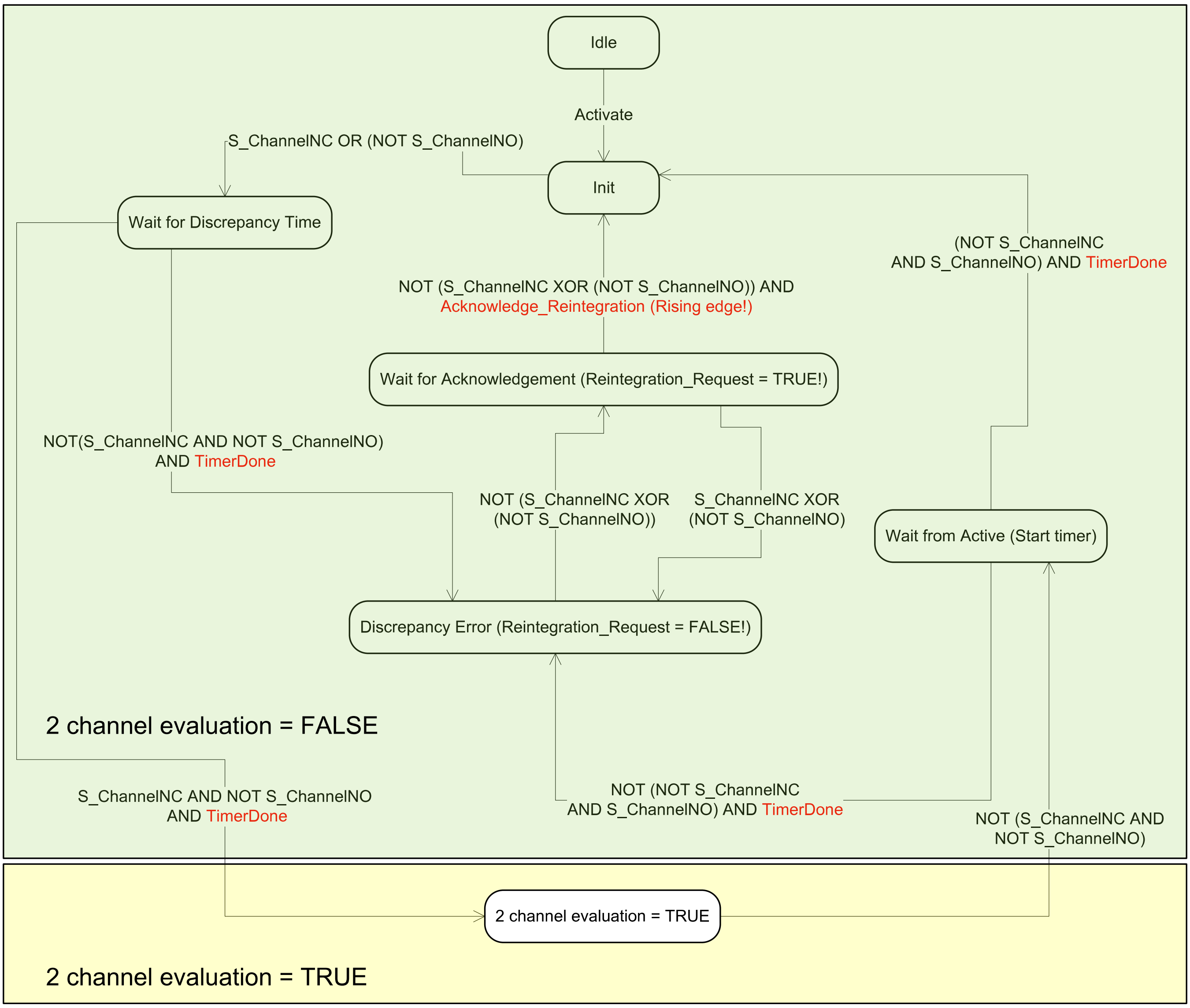


Fig. 19: 2 channel antivalent mode implemented in DI581-S

> **NOTICE!**
> 2 channel equivalent and 2 channel antivalent modes are implemented in DI581-S and DX581-S module to handle relatively static safety signals, e.g., those for emergency stop devices.
>
> If frequently changing signals, like those from light curtains, laser scanners, door switches, etc. must be handled by DI581-S and DX581-S, then it is highly recommended to use input delay of 1 ms for these channels or to configure related channels in 1 channel mode and do 2 channel equivalent and 2 channel antivalent evaluation at the safety CPU using PLCopen Safety FBs SF_Equivalent *Chapter 4.6.4.2 “SF_Equivalent” on page 202* and SF_Antivalent *Chapter 4.6.4.3 “SF_Antivalent” on page 207*.

3.3.3 Mounting, dimensions and electrical connection

The input modules can be plugged only on spring-type TU582-S I/O terminal unit. The unique mechanical coding on I/O terminal units prevents a potential mistake of placing the non-safety I/O module on safety I/O terminal unit and the other way around. Basic information on system assembly is shown here. Detailed information can be found in *[4]*.

Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204 part 1, by skilled electricians only.

Assembly of DI581-S

> **DANGER!**
> Hot plug and hot swap of energized modules is not permitted. All power sources (supply and process voltages) must be switched off while working with safety modules.

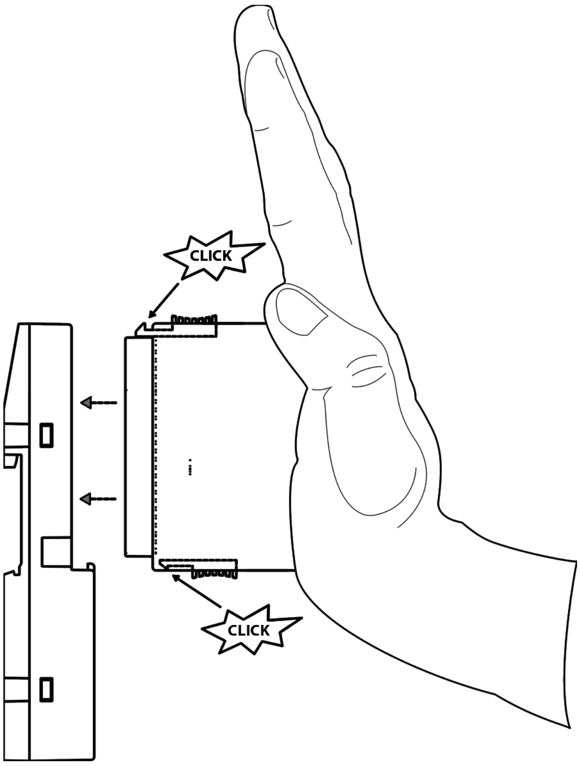


Fig. 20: Assembly instructions

1. Put the module on the terminal unit.

   ⇨ The module clicks in.

2. Then press the module with a force of at least 100 N into the terminal unit to achieve proper electrical contact.

Disassembly of DI581-S

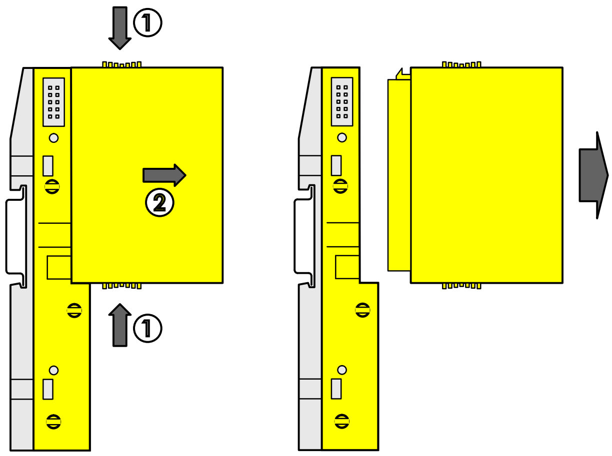


Fig. 21: Disassembly instructions

▷ Press above and below, then remove the module.

Dimensions

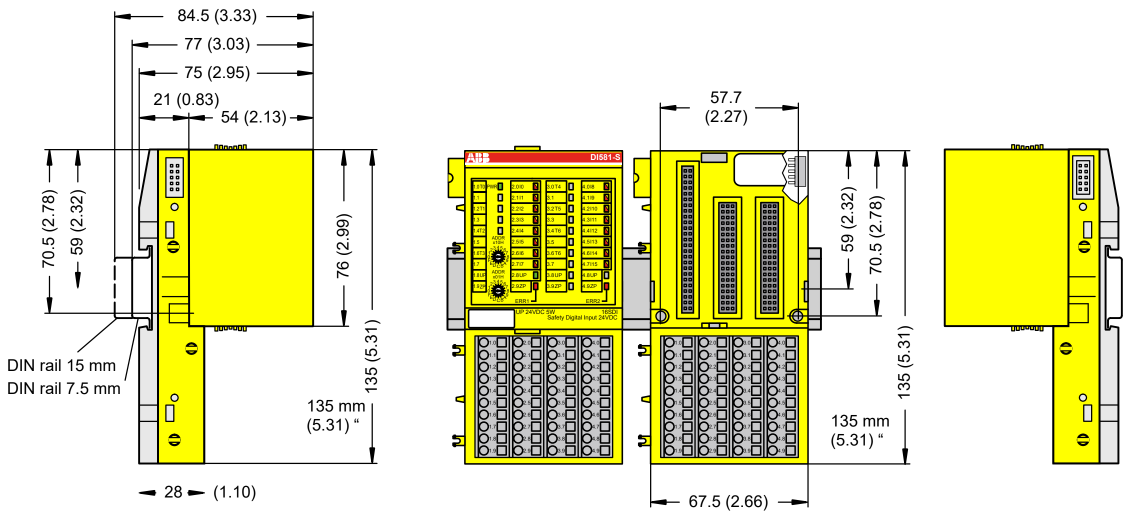


Fig. 22: Dimensions of DI581-S safety I/O module

Electrical connection

> **NOTICE!**
> The same TU582-S is used by all AC500-S safety I/O modules. If TU582-S is wired for DX581-S module with safety digital outputs and DI581-S or AI581-S modules are occasionally placed on this terminal unit, under no circumstances it is possible that safety digital output clamps on TU582-S become energized due to a wrongly placed DI581-S or AI581-S safety I/O modules.

The electrical connection of the I/O channels is carried out using 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8, 2.8, 3.8 and 4.8 as well as 1.9, 2.9, 3.9 and 4.9 are electrically interconnected within the I/O terminal unit and have always the same assignment, independent of the inserted module:

- Terminals 1.8, 2.8, 3.8 and 4.8: Process voltage UP = +24 V DC
- Terminals 1.9, 2.9, 3.9 and 4.9: Process voltage ZP = 0 V

The assignment of other terminals:

Terminals	Signal	Meaning
1.0, 1.2, 1.4, 1.6, 3.0, 3.2, 3.4, 3.6	T0, T1, T2, T3, T4, T5, T6, T7	Connectors of 8 test pulse outputs T0, T1, T2, T3, T4, T5, T6, T7
2.0 ... 2.7, 4.0 ... 4.7	I0, I1, I2, I3, I4, I5, I6, I7, I8, I9, I10, I11, I12, I13, I14, I15	16 safety digital inputs
1.8, 2.8, 3.8, 4.8	UP	Process power supply +24 V DC
1.9, 2.9, 3.9, 4.9	ZP	Central process earth
1.1, 1.3, 1.5, 1.7, 3.1, 3.3, 3.5, 3.7	Free	Not used

NOTICE!
The process voltage must be included in the earthing concept of the control system (e.g., earthing the minus pole).

Examples of connections

Examples of electrical connections with DI581-S module and single channel Ix.

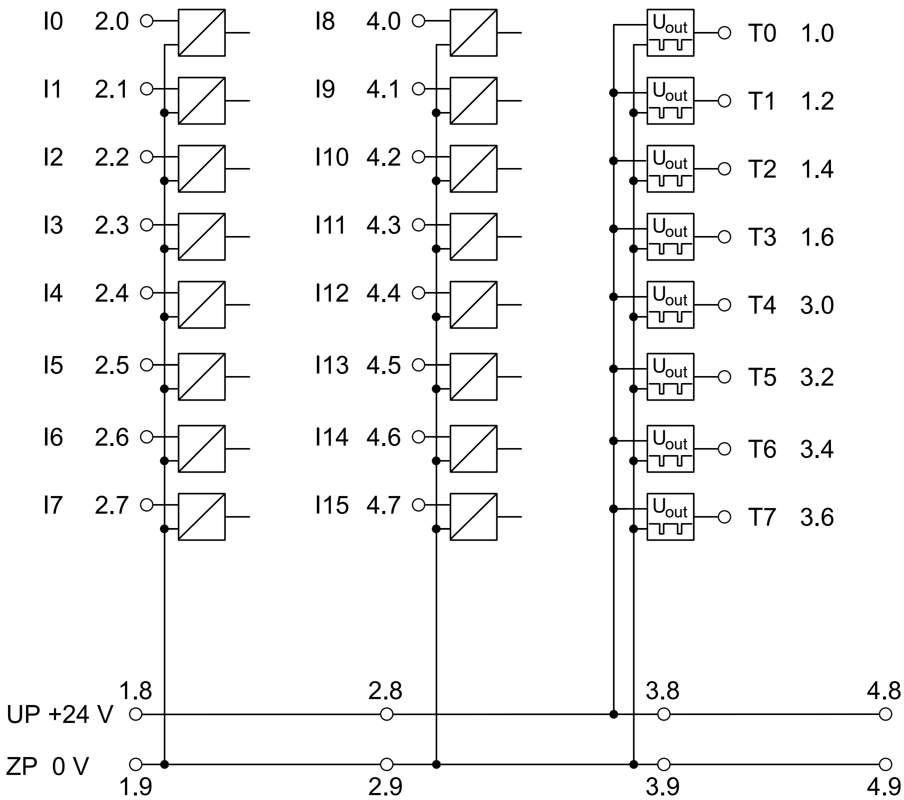


Fig. 23: Example of electrical connections with DI581-S

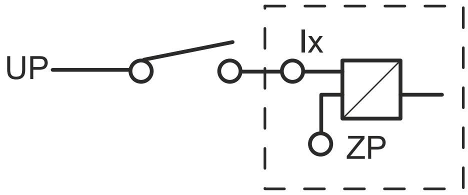


Fig. 24: Example of single channel with DI581-S

3.3.4 Internal data exchange

Inputs (bytes)	6
Outputs (bytes)	2

3.3.5 I/O configuration

The safety digital input module DI581-S does not store configuration data itself. The configuration data is stored on the safety and non-safety CPUs.

3.3.6 Parameterization

The arrangement of the parameter data is performed by your system configuration software Automation Builder. ABB GSDML file for PROFINET devices can be used to configure DI581-S parameters in 3rd party PROFINET F-Host systems.

The parameter setting directly influences the functionality of modules and reachable SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1).

No.	Name	Values	Default
1	Check supply	"On", "Off"	"On"
2	Configuration	"Not used", "1 channel", "2 channel equivalent", "2 channel antivalent"	"Not used"
3	Test pulse	"Disabled", "Enabled"	"Disabled"
4	Input delay	"1 ms", "2 ms", "5 ms", "10 ms", "15 ms", "30 ms", "50 ms", "100 ms", "200 ms", "500 ms"	"5 ms"
5	Discrepancy time*	"10 ms", "20 ms", "30 ms", "40 ms", "50 ms", "60 ms", "70 ms", "80 ms", "90 ms", "100 ms", "150 ms", "200 ms", "250 ms", "300 ms", "400 ms", "500 ms", "750 ms", "1 s", "2 s", "3 s", "4 s", "5 s", "10 s", "20 s", "30 s"	"50 ms"

* Available only for "2 channel equivalent" and "2 channel antivalent" configuration

3.3.7 Circuit examples DI581-S

Examples of electrical connections and reachable SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1) with DI581-S module are presented below.

NOTICE!

Whenever DC = High is used in the circuit examples with safety digital inputs, the following measure from ISO 13849-1 *[10]* is used with DI581-S module: Cross monitoring of input signals and intermediate results within the logic (L), and temporal and logical software monitor of the program flow and detection of static faults and short circuits (for multiple I/O).

Whenever DC = Medium is used in the circuit examples with safety digital inputs, any of the measures for input devices with DC ≥ 90 % can be used from ISO 13849-1 *[10]*.

1-channel sensor

SILCL/ PL [1), 2)]	SILCL 1 / PL c
SIL[3)]	SIL 2

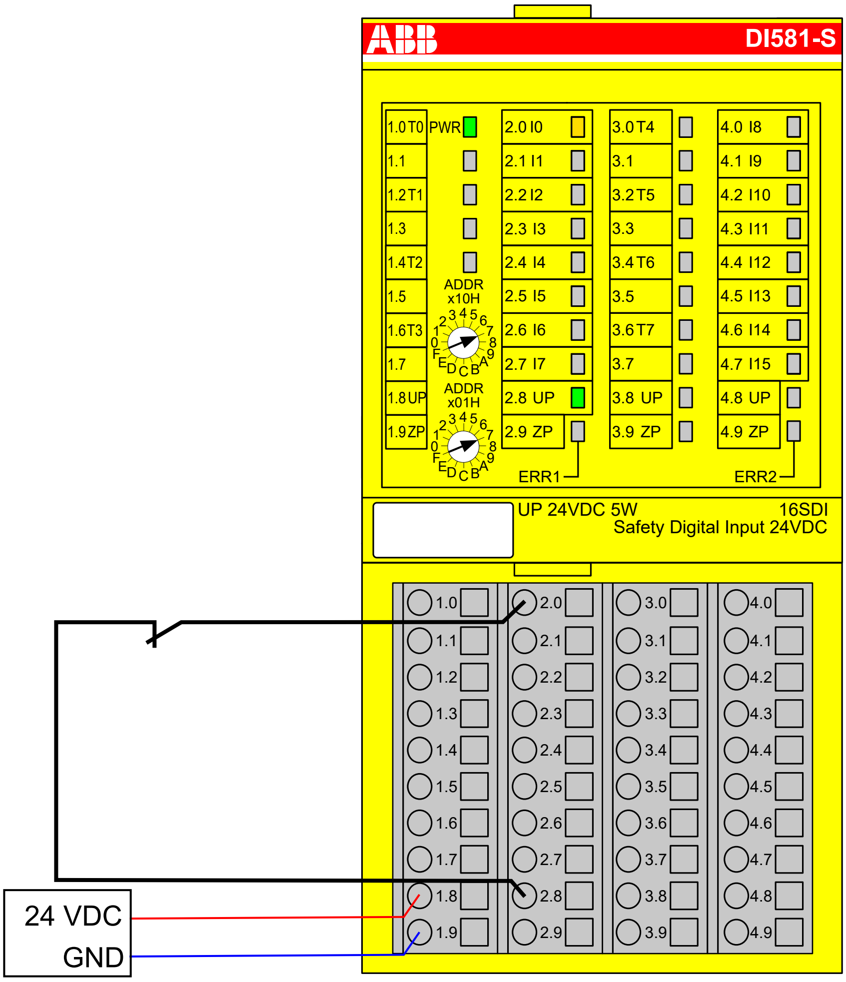


Fig. 25: Circuit example DI581-S, 1-channel sensor

1) - MTTFd = High, DC = 0

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher levels up to SIL 3 with error exclusion)

1-channel OSSD output (with internal tests)

SILCL / PL [1), 2)]	SILCL 1 / PL c
SIL[3)]	SIL 2

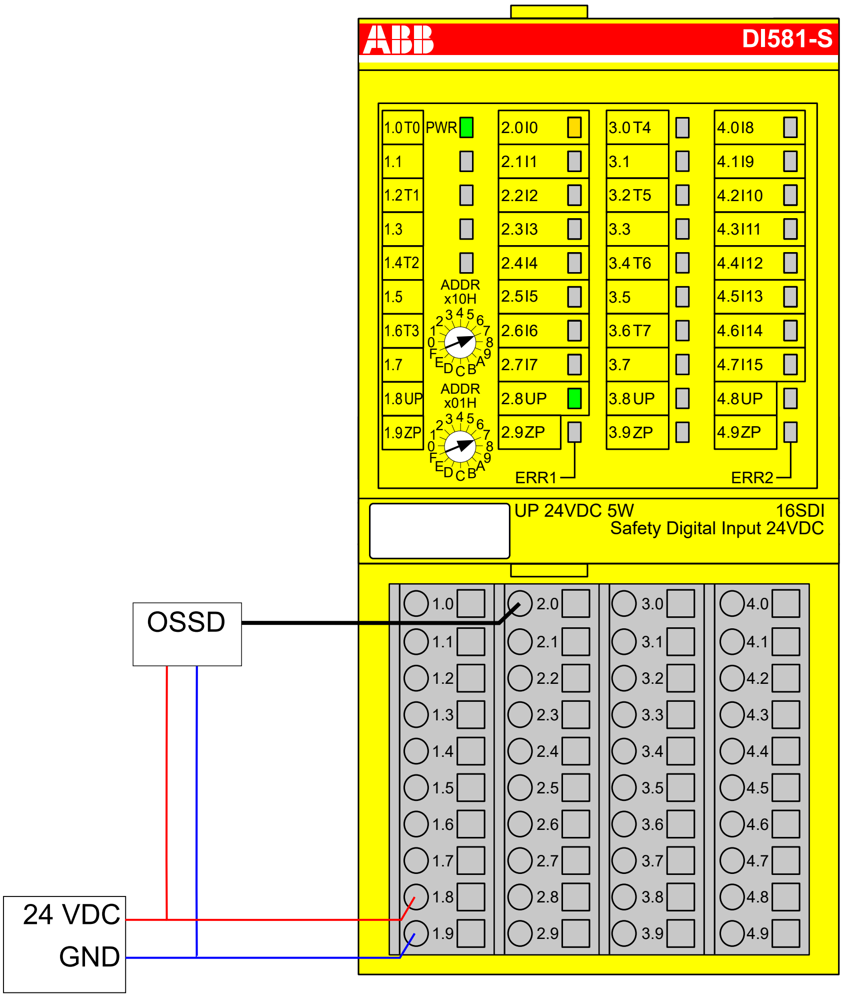


Fig. 26: Circuit example DI581-S, 1-channel OSSD output (with internal tests)

1) - MTTFd = High, DC = 0

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher levels up to SIL 3 with error exclusion)

2-channel sensor (equivalent)

2-channel evaluation	In DI581-S module
SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 3

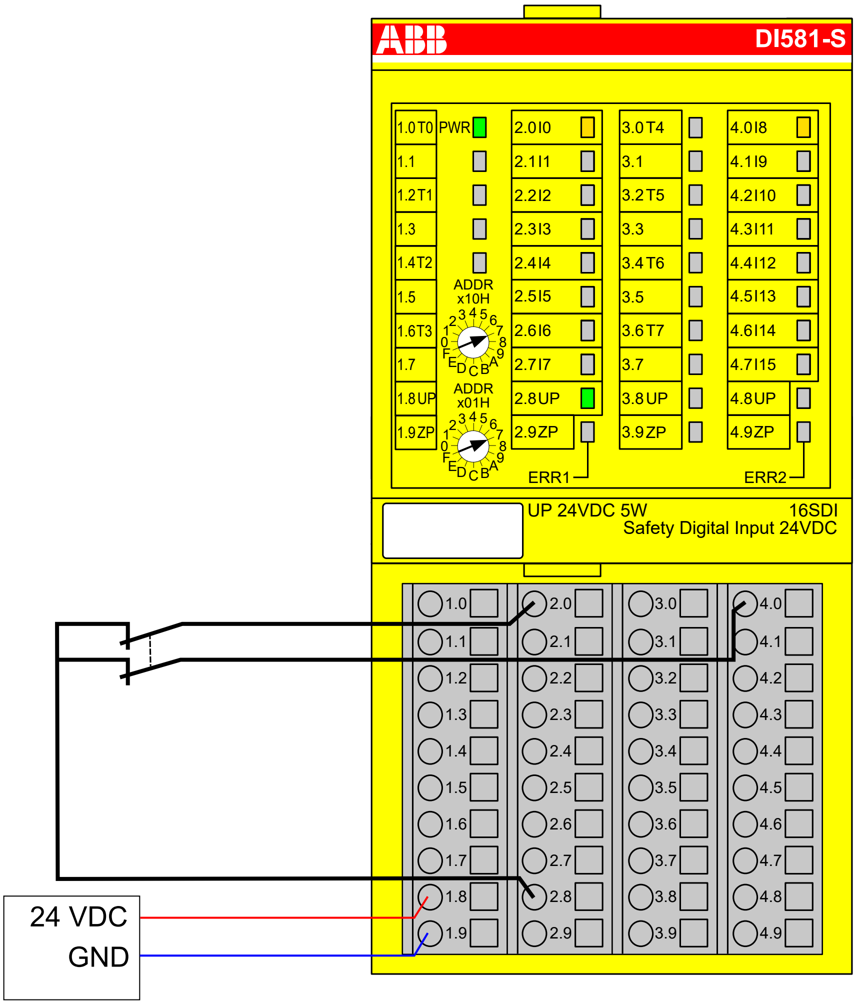


Fig. 27: Circuit example DI581-S, 2-channel sensor (equivalent)

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

2-channel sensor (antivalent)

2-channel evaluation	In DI581-S module
SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 3

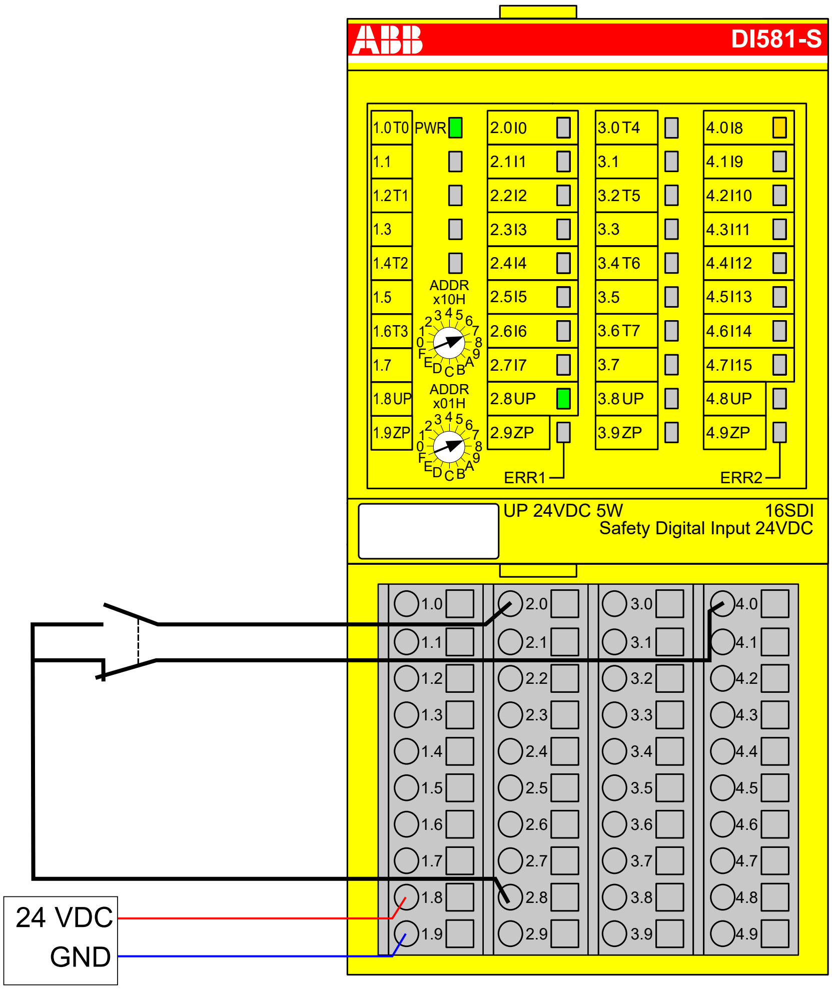


Fig. 28: Circuit example DI581-S, 2-channel sensor (antivalent)

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

2-channel OSSD output (with internal tests)

2-channel evaluation	In DI581-S module
SILCL / PL [1), 2)]	SILCL 3 / PL e
SIL[3)]	SIL 3

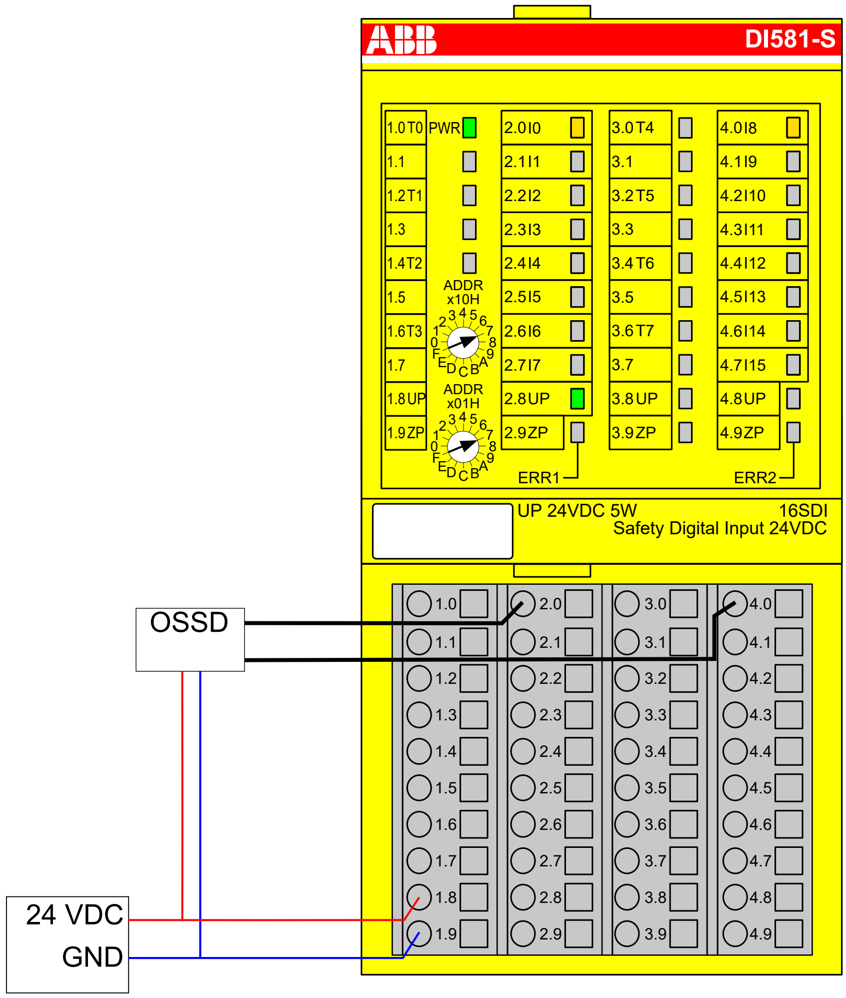


Fig. 29: Circuit example DI581-S, 2-channel OSSD output (with internal tests)

1) - MTTFd = High, DC = High

2) - Max. reachable (ISO 13849-1, IEC 62061)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

1-channel sensor with test pulses

SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 3

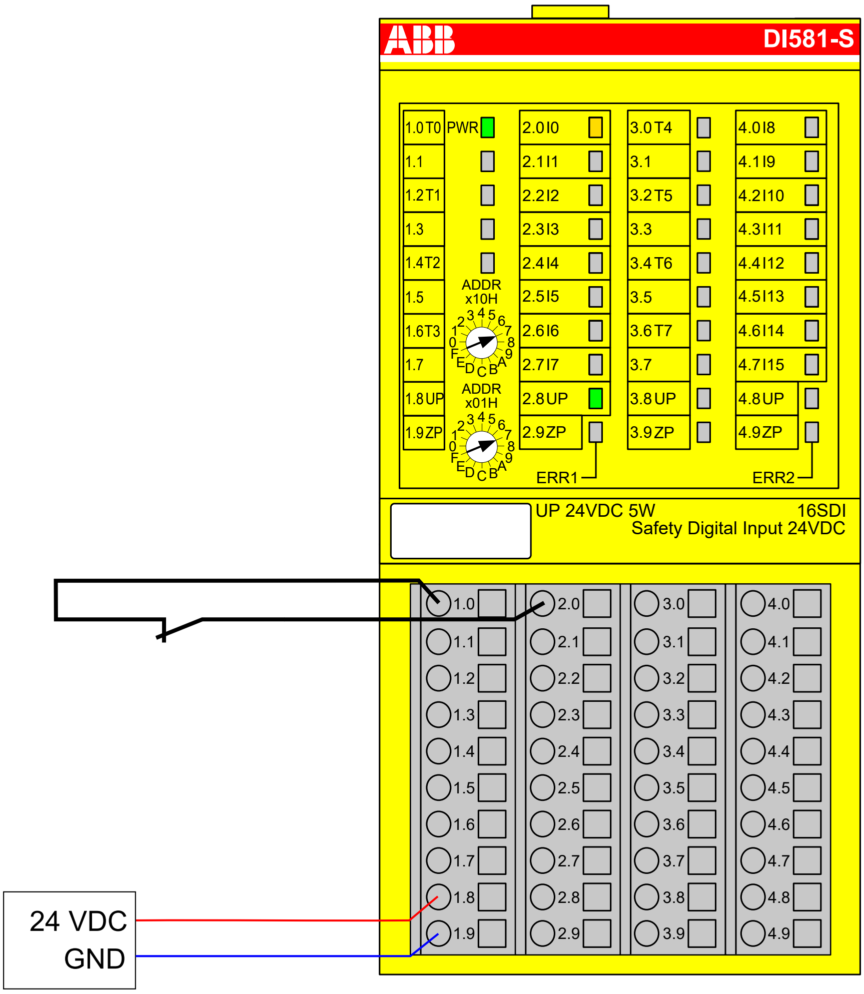


Fig. 30: Circuit example DI581-S, 1-channel sensor with test pulses

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

2-channel sensor (equivalent) with test pulses

2-channel evaluation	In safety CPU
SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 3

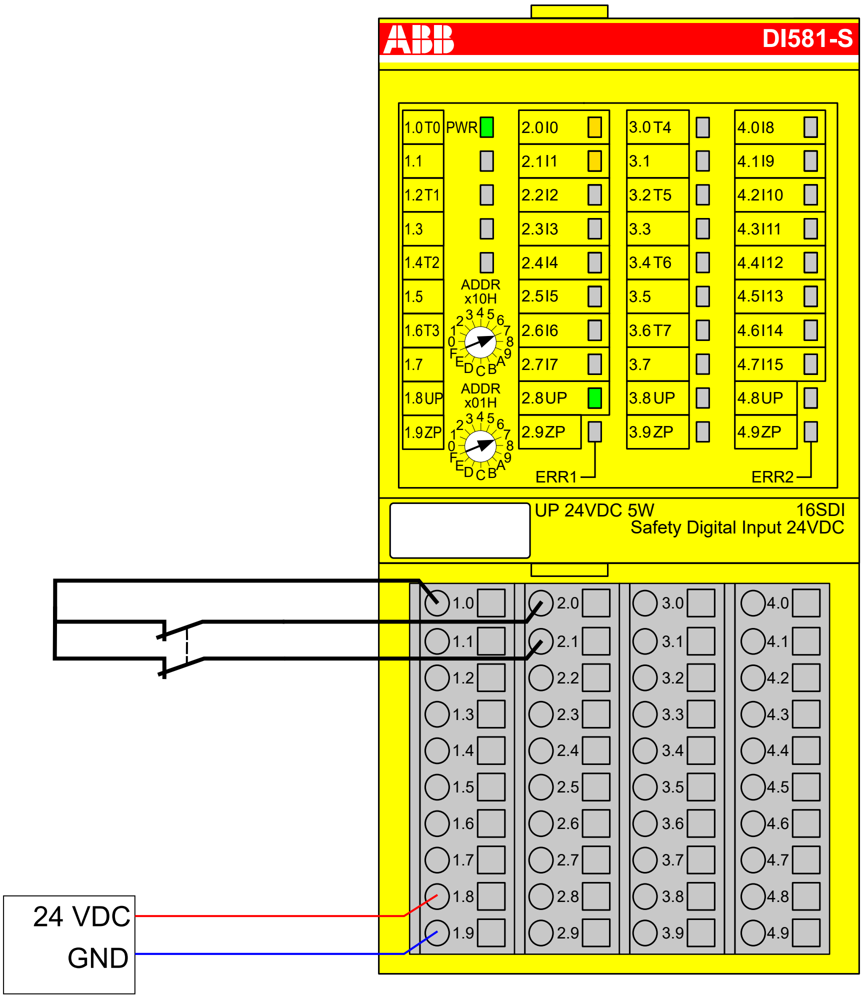


Fig. 31: Circuit example DI581-S, 2-channel sensor (equivalent) with test pulses

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

2-channel sensor (equivalent) with test pulses

2-channel evaluation	In DI581-S module
SILCL / PL [1), 2)]	SILCL 3 / PL e
SIL[3)]	SIL 3

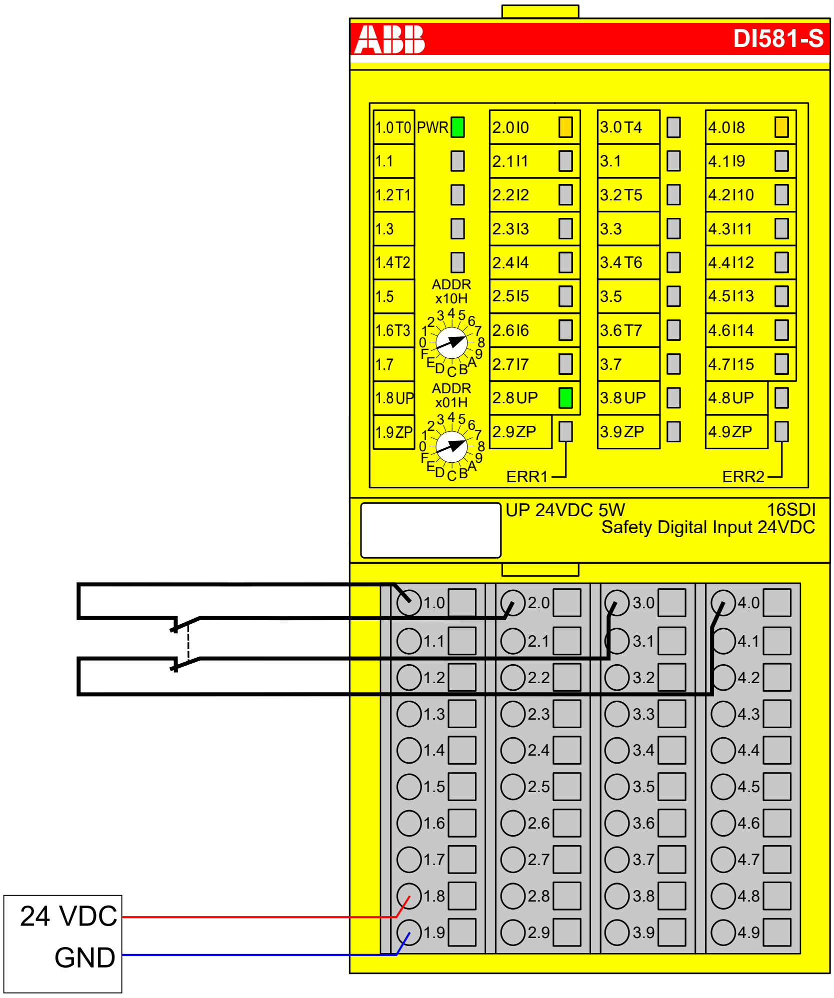


Fig. 32: Circuit example DI581-S, 2-channel sensor (equivalent) with test pulses

1) - MTTFd = High, DC = High

2) - Max. reachable (ISO 13849-1, IEC 62061)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

2 x OSSD output (with internal tests)

2-channel evaluation	In DI581-S module
SILCL / PL [1), 2)]	SILCL 3 / PL e
SIL[3)]	SIL 3

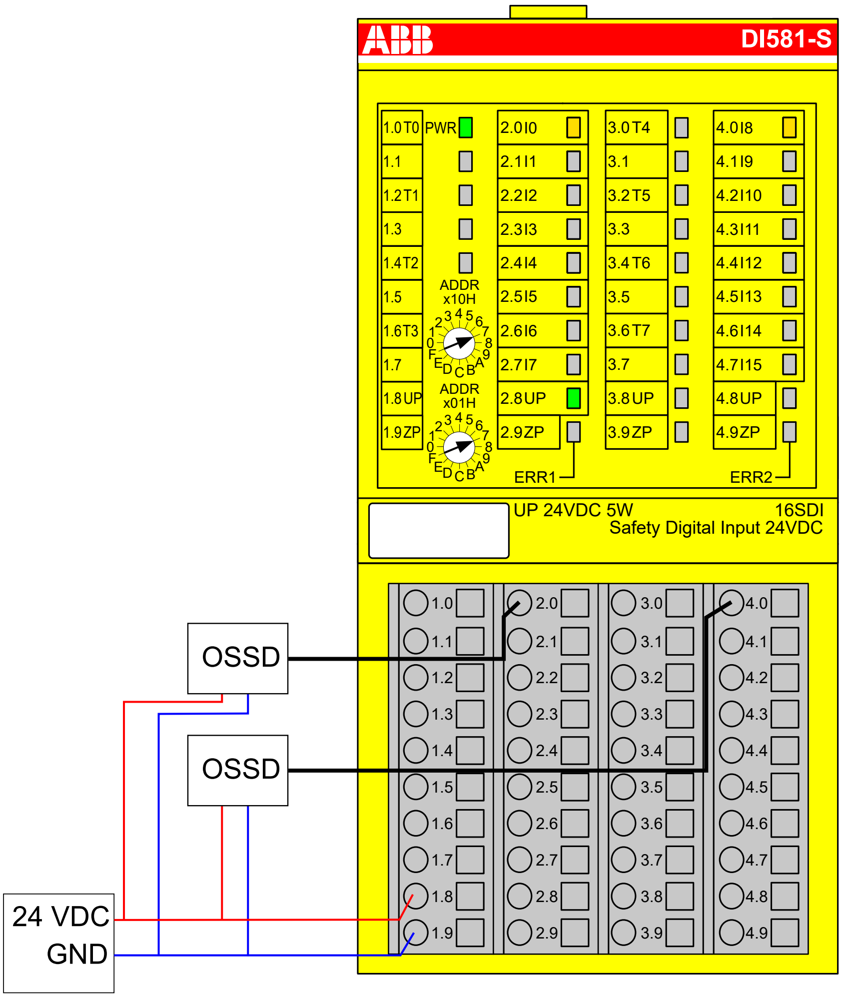


Fig. 33: Circuit example DI581-S, 2 x OSSD output (with internal tests)

1) - MTTFd = High, DC = High

2) - Max. reachable (ISO 13849-1, IEC 62061)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

2 separate sensors with test pulses

2-channel evaluation	In safety CPU
SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 3

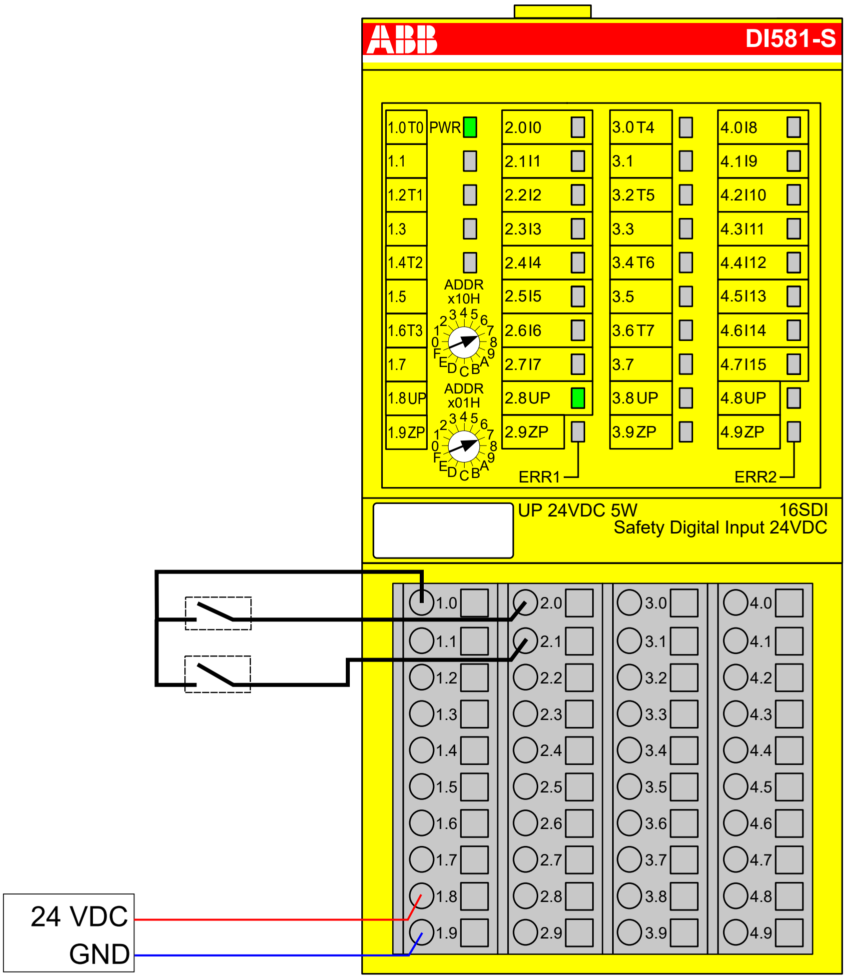


Fig. 34: Circuit example DI581-S, 2 separate sensors with test pulses

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

2 x 2-channel sensor (antivalent) with test pulses

2-channel evaluation	First in DI581-S module and then in the safety CPU
SILCL / PL [1), 2)]	SILCL 3 / PL e
SIL[3)]	SIL 3

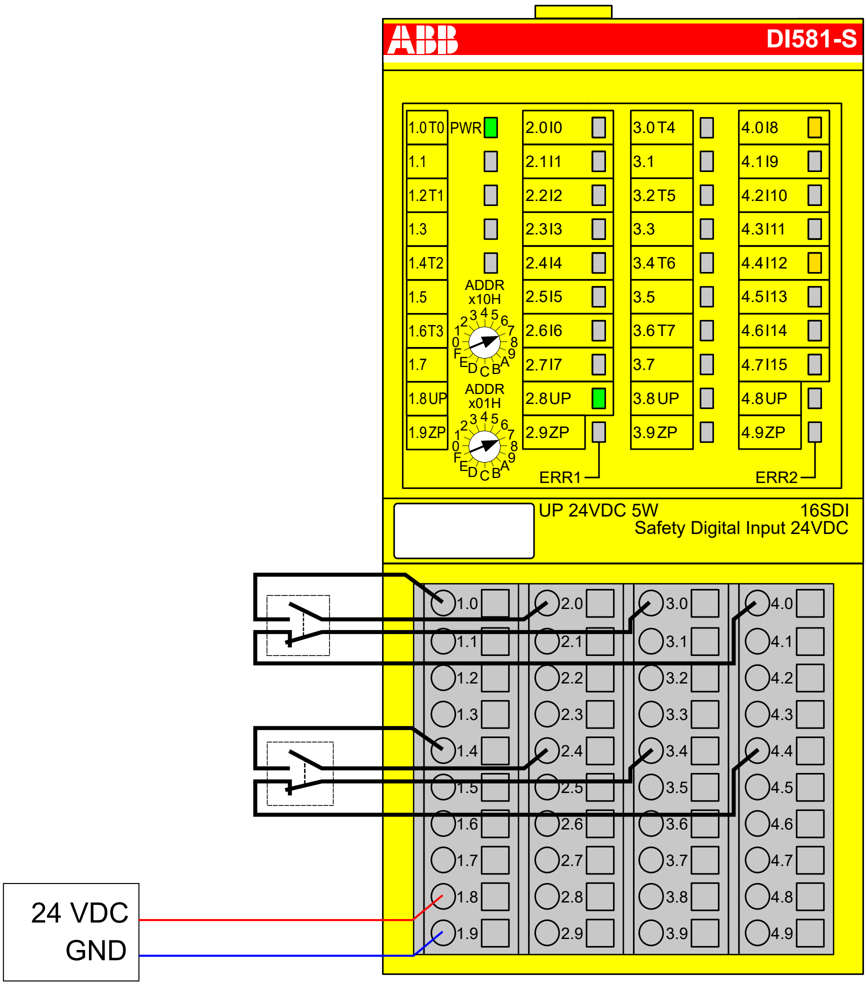


Fig. 35: Circuit example DI581-S, 2 x 2-channel sensor (antivalent) with test pulses

1) - MTTFd = High, DC = High

2) - Max. reachable (ISO 13849-1, IEC 62061)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

Mode switch 1 from 4

Mode switch evaluation	In safety CPU
SILCL / PL [1), 2)]	SILCL 1 / PL c
SIL[3)]	SIL 2

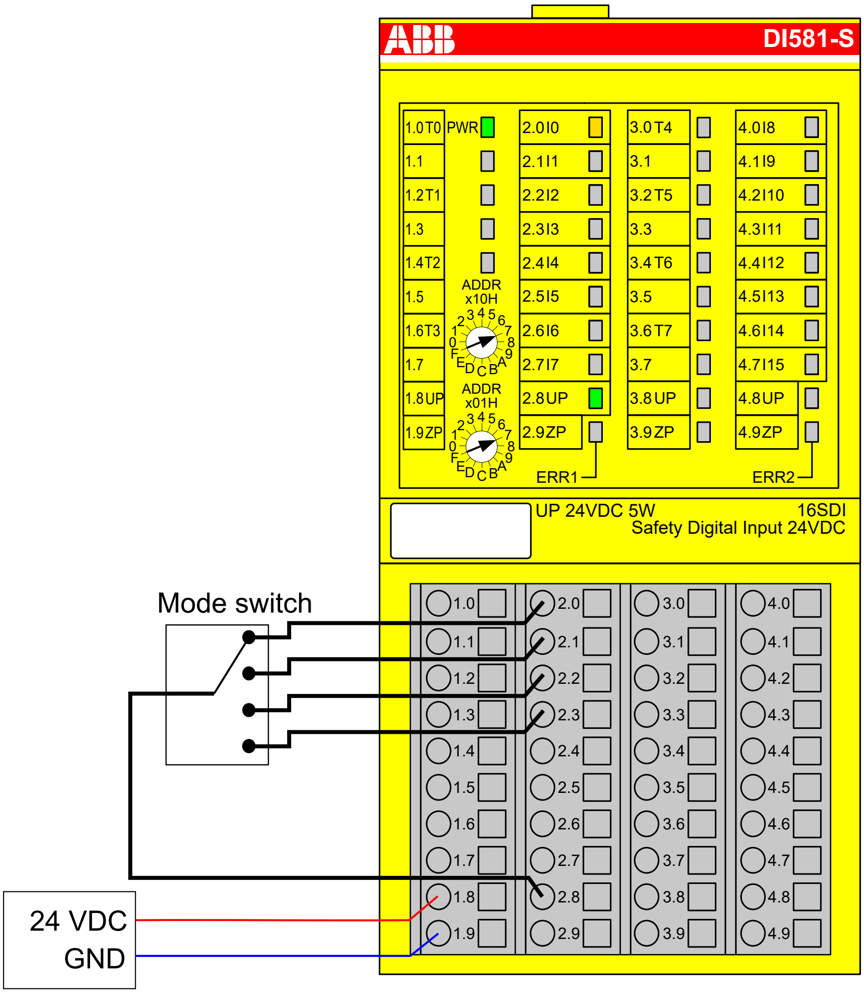


Fig. 36: Circuit example DI581-S, mode switch 1 from 4

1) - MTTFd = High, DC = Low

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher levels up to SIL 3 with error exclusion)

3.3.8 LED status display

Table 5: Status display and its meaning

LED	Description	Color	LED = OFF	LED = ON	LED flashes
Inputs 0 ... 15	Digital input	Yellow	Input = OFF	Input = ON (the input voltage is displayed even if the supply voltage is OFF).	-
	Channel error	Red	No channel error	Channel error	-

LED	Description	Color	LED = OFF	LED = ON	LED flashes
UP	Process voltage +24 V DC via terminal	Green	Process supply voltage is missing	Process supply voltage is OK	-
PWR	+3.3 V voltage from I/O bus	Green	+3.3 V I/O bus voltage is not available	+3.3 V I/O bus voltage is available	-
ERR1	Module error indicator 1	Red	No module error	Module error which leads to a SAFE STOP state	Module passivation and/or acknowledgment request (alternating blinking)
ERR2	Module error indicator 2	Red			

3.3.9 Technical data

> **NOTICE!**
> DI581-S-XC version is available for usage in extreme environmental conditions ↬ *Appendix A “System data for AC500-S-XC” on page 367*.

Additional technical data is available in ABB PLC catalog at *www.abb.com/plc*.

Process supply voltage UP

Data	Value	Unit
Connections terminals 1.8 ... 4.8 (UP)	+24	V
Connections terminals 1.9 ... 4.9 (ZP)	0	V
Rated value (-15 %, +20 %, without ripple)	24	V DC
Max. ripple	5	%
Protection against reversed voltage	Yes	
Rated protection fuse for UP (fast)	10	A
Electrical isolation	per module	
Mechanisms in which I/Os are processed	periodically refreshed	
Current consumption from UP at normal operation with + 24 V DC (for module electronics)	0.18	A
Inrush current from UP at 30 V (at power up)	0.1	A^2s
Inrush current from UP at 24 V (at power up)	0.06	A^2s

> **NOTICE!**
> All DI581-S channels (including test pulse outputs) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 V DC.

Mounting position

Horizontal or vertical with derating (maximal operating temperature reduced to +40 °C)

Cooling

The natural convection cooling must not be hindered by cable ducts or other parts in the switchgear cabinet.

Allowed interruptions of power supply, according to EN 61131-2

Data	Value	Unit
DC supply interruptions	< 10	ms
Time between 2 DC supply interruptions, PS2	> 1	s

Environmental conditions

Data	Value	Unit
Operating temperature*	0 ... +60	°C
Storage temperature	-40 ... +85	°C
Transport temperature	-40 ... +85	°C
Humidity without condensation	max. 95	%
Operating air pressure	> 800	hPa
Storage air pressure	> 660	hPa
Operating altitude	< 2000	m above sea level
Storage altitude	< 3500	m above sea level

* Extended temperature ranges (below 0 °C and above +60 °C) can be supported in special versions of DI581-S *Appendix A “System data for AC500-S-XC” on page 367*.

Creepage distances and clearances

The creepage distances and clearances meet the overvoltage category II, pollution degree 2.

Power supply units

For the supply of modules, power supply units according to PELV/SELV specifications must be used.

Electromagnetic compatibility

For information on electromagnetic compatibility refer to the latest TÜV SÜD Report *[2]*.

Mechanical properties

Data	Value	Unit
Degree of protection	IP 20	
Housing	according to UL 94	
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 3.5 mm	2 ... 15	Hz
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 1 g *	15 ...150	Hz
Shock test (all three axes), 11 ms half-sinusoidal	15	g
MTBF	102	years

* Higher values on request

Self-test and diagnostic functions

Start-up and runtime tests: Program flow control, RAM, CPU, cross-talk, stuck-at-1, etc.

Dimensions, weight

Data	Value	Unit
W x H x D	67.5 x 76 x 62	mm
Weight	~ 130	g

Certifications CE, cUL (further certifications at *www.abb.com/plc*)

3.3.9.1 Technical data of safety digital inputs

Data	Value	Unit
Number of input channels per module	16	
Terminals of the channels I0 to I7	2.0 ... 2.7	
Terminals of the channels I8 to I15	4.0 ... 4.7	
Terminals of reference potential for all inputs (minus pole of the process supply voltage, signal name ZP)	1.9 ... 4.9	
Electrical isolation from the rest of the module (I/O bus)	Yes	
Input type acc. to EN 61131-2	Type 1	
Input delay (0 ➔ 1 or 1 ➔ 0), configurable	1 ... 500	ms

Input signal indication One yellow LED per channel, the LED is ON when the input signal is high (signal 1).

Signal voltage

Data	Value	Unit
Input signal voltage	24	V DC
Signal 0	-3 ... +5	V
Undefined signal	> +5 ... < +15	V
Signal 1	+15 ... +30	V

Input current per channel

Data	Value	Unit
Input voltage +24 V, typically	7	mA
Input voltage +5 V	> 1	mA
Input voltage +15 V	> 4	mA
Input voltage +30 V	< 8	mA

Cable length

Data	Value	Unit
Max. cable length, shielded	1000	m
Max. cable length, unshielded	600	m

3.3.9.2 Technical data of non-safety test pulse outputs

DANGER!
Exceeding the permitted process or supply voltage range (< -35 V DC or > +35 V DC) could lead to unrecoverable damage of the system.

Data	Value	Unit
Number of test pulse channels per module (transistor test pulse outputs)	8	
Terminals of the channels T0 to T3	1.0, 1.2, 1.4, 1.6	
Terminals of the channels T4 to T7	3.0, 3.2, 3.4, 3.6	
Terminals of reference potential for all test pulse outputs (minus pole of the process supply voltage, signal name ZP)	1.9 ... 4.9	
Terminals of common power supply voltage for all outputs (plus pole of the process supply voltage, signal name UP)	1.8 ... 4.8	
Output voltage for signal 1	UP - 0.8	V
Length of test pulse 0 phase	1	ms

Output current

Data	Value	Unit
Rated value, per channel	10	mA
Maximum value (all channels together)	80	mA
Short-circuit proof / overload proof	yes	
Output current limitation	65	mA
Resistance to feedback against 24V signals	yes	

Cable length

Data	Value	Unit
Max. cable length, shielded	1000	m
Max. cable length, unshielded	600	m

3.3.10 Ordering data

Type	Description	Part no.
DI581-S	Safety digital input module 16SDI	1SAP 284 000 R0001
DI581-S-XC	Safety digital input module 16SDI, extreme conditions	1SAP 484 000 R0001

3.4 DX581-S safety digital input/output module

Elements of the module

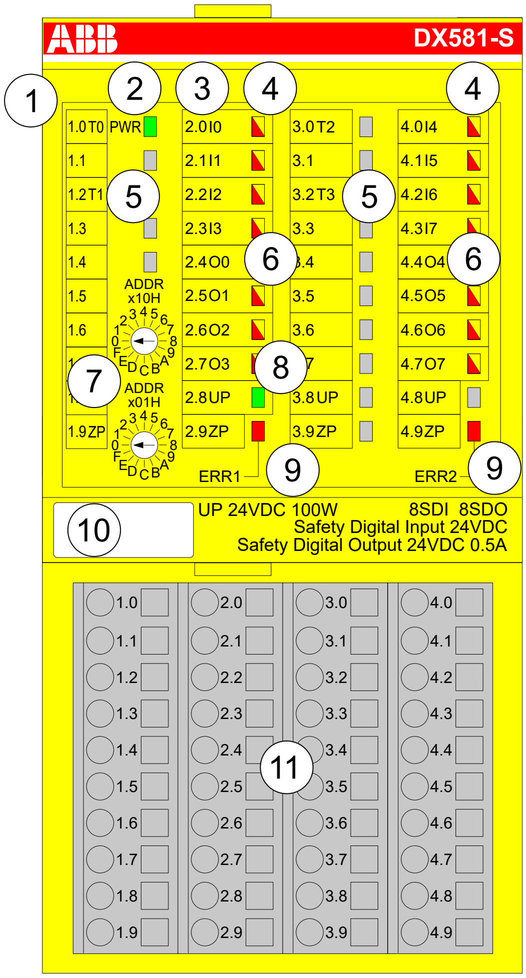


Fig. 37: Safety digital input/output module DX581-S, plugged on terminal unit TU582-S

1 I/O bus
2 System LED
3 Allocation terminal no. - signal name
4 8 yellow/red LEDs signal status I0 ... I3/I4 ... I7
5 4 test pulse outputs T0 ... T1/T2 ... T3
6 8 yellow/red LEDs signal status O0 ... O3 / O4 ... O7
7 2 rotary switches for PROFIsafe address
8 Green LED for process voltage UP
9 Red LEDs to display module errors
10 Label (TA525)
11 I/O terminal unit (TU582-S)

3.4.1 Purpose

Safety digital input/output module DX581-S can be used as a remote expansion module at CI501-PNIO, CI502-PNIO, CI504-PNIO and CI506-PNIO PROFINET modules or locally at AC500 CPUs for up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1) safety applications.

> **NOTICE!**
> SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1) reachable in your safety application depend on the wiring of your sensors and actuators to DX581-S module *Chapter 3.4.7 “Circuit examples DX581-S” on page 98*.

DX581-S contains 8 safety digital inputs 24 V DC separated in two groups (2.0 ... 2.3 and 4.0 ... 4.3) and 8 safety digital transistor outputs with no potential separation between the channels.

The inputs/outputs are not electrically isolated from the other electronic circuitry of the module.

3.4.2 Functionality

Digital inputs	8 (24 V DC)
Digital outputs	8 (24 V DC)
LED displays	for signal status, module errors, channel errors and supply voltage
Internal power supply	through the expansion bus interface (I/O bus)
External power supply	via the terminals ZP and UP (process voltage 24 V DC)

Self-tests and diagnostic functions (both start-up and runtime), like CPU and RAM tests, program flow control, cross-talk and stuck-at-1 tests, etc. are implemented in DX581-S according to IEC 61508 SIL 3 requirements.

> **NOTICE!**
> Only F_Dest_Add is used for PROFIsafe F-Device identification in DX581-S.

DX581-S contains 8 safety digital input channels with the following features:

- Phase-shifted (unique) test pulses T0 ... T3 can be used for connection of mechanical sensors. Test pulse outputs T0 ... T3 provide 24 V signal with a short phase-shifted unique pulses (0 V) of 1 ms. Since the test pulses on each of the test pulse output channels are unique (due to the phase shift), they can be used to monitor the cross-talk between the given input channel with connected test pulse output and another wire, e.g, with 24 V DC, another test pulse output, etc. Test pulse outputs are dedicated ones:
  – T0 can be used only with input channels I0 and I1
  – T1 can be used only with input channels I2 and I3
  – T2 can be used only with input channels I4 and I5
  – T3 can be used only with input channels I6 and I7
- Input delay with the following values: 1 ms, 2 ms, 5 ms, 10 ms, 15 ms, 30 ms, 50 ms, 100 ms, 200 ms, 500 ms. Input delay value of 1 ms is the minimum one.

> **NOTICE!**
> The allowed signal frequency on safety digital inputs is dependent on the input delay value for the given channel:
> – For channel input delay values of 1 ... 10 ms, the pulse length of input signal shall be ≥ 15 ms (~ 65 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 15 ms, the pulse length of input signal shall be ≥ 20 ms (~ 50 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 30 ms, the pulse length of input signal shall be ≥ 40 ms (~ 25 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 50 ms, the pulse length of input signal shall be ≥ 60 ms (~ 15 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 100 ms, the pulse length of input signal shall be ≥ 120 ms (~ 8 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 200 ms, the pulse length of input signal shall be ≥ 250 ms (~ 4 Hz) to avoid occasional input channel passivation.
> – For channel input delay of 500 ms, the pulse length of input signal shall be ≥ 600 ms (~ 1.5 Hz) to avoid occasional input channel passivation.

DANGER!

The input delay parameter means that signals with the duration shorter than input delay value are always not captured by the safety module.

The signals with the duration of equal to or longer than "input delay parameter" + "input delay accuracy" are always captured by the safety module, provided that the allowed frequency (refer to previous notice) of the safety input signal is not exceeded.

The "input delay accuracy" can be estimated based on the following assumptions:

– If no test pulses are configured for the given safety digital input, then input delay accuracy can be calculated as 1 % of set input delay value (however, input delay accuracy value must be at least 0.5 ms!).
– If test pulses are configured for the given safety digital input of DX581-S module, then the input delay accuracy values can be estimated based on the input delay parameter value *Table 6 "Input delay accuracy for DX581-S" on page 90*.

Table 6: Input delay accuracy for DX581-S

Input delay (ms)	**Input delay accuracy (ms)**
1	2
2	2
5	3
10	4
15	5
30	6
50	10
100	15
200	25
500	50

- Checking of process power supply (diagnostic message is sent from the safety I/O module to the CPU informing about the lack of process power supply for the given safety I/O module). This function is a non-safety one and is not related to the internal safety-relevant over- and undervoltage detection.
- 2 channel equivalent and 2 channel antivalent mode with discrepancy time monitoring (configurable 10 ms ... 30 s).

NOTICE!

In a 2 channel mode, the lower channel (channels 0/4 ➔ Channel 0, channels 1/5 ➔ Channel 1, etc.) transports the aggregated process value, PROFIsafe diagnostic bit, acknowledgment request and acknowledge reintegration information. The higher channel always provides the passivated value "0".

DANGER!

After discrepancy time error, the relevant channels are passivated. As soon as a valid sensor state is observed (equivalent or antivalent, depending on the selected mode), reintegration request status bit for the given channel becomes TRUE. You can acknowledge an error using acknowledge reintegration command bit for the given channel. This can directly lead to the machine start, because both TRUE - TRUE and FALSE - FALSE are valid states for equivalence and TRUE - FALSE and FALSE - TRUE are valid states for antivalence.

Make sure that such behavior is acceptable in your safety application. If no, then you can use either included PLCopen Safety POUs for 2 channel evaluation in your safety program or write your own POUs for 2 channel evaluation on the safety CPU.

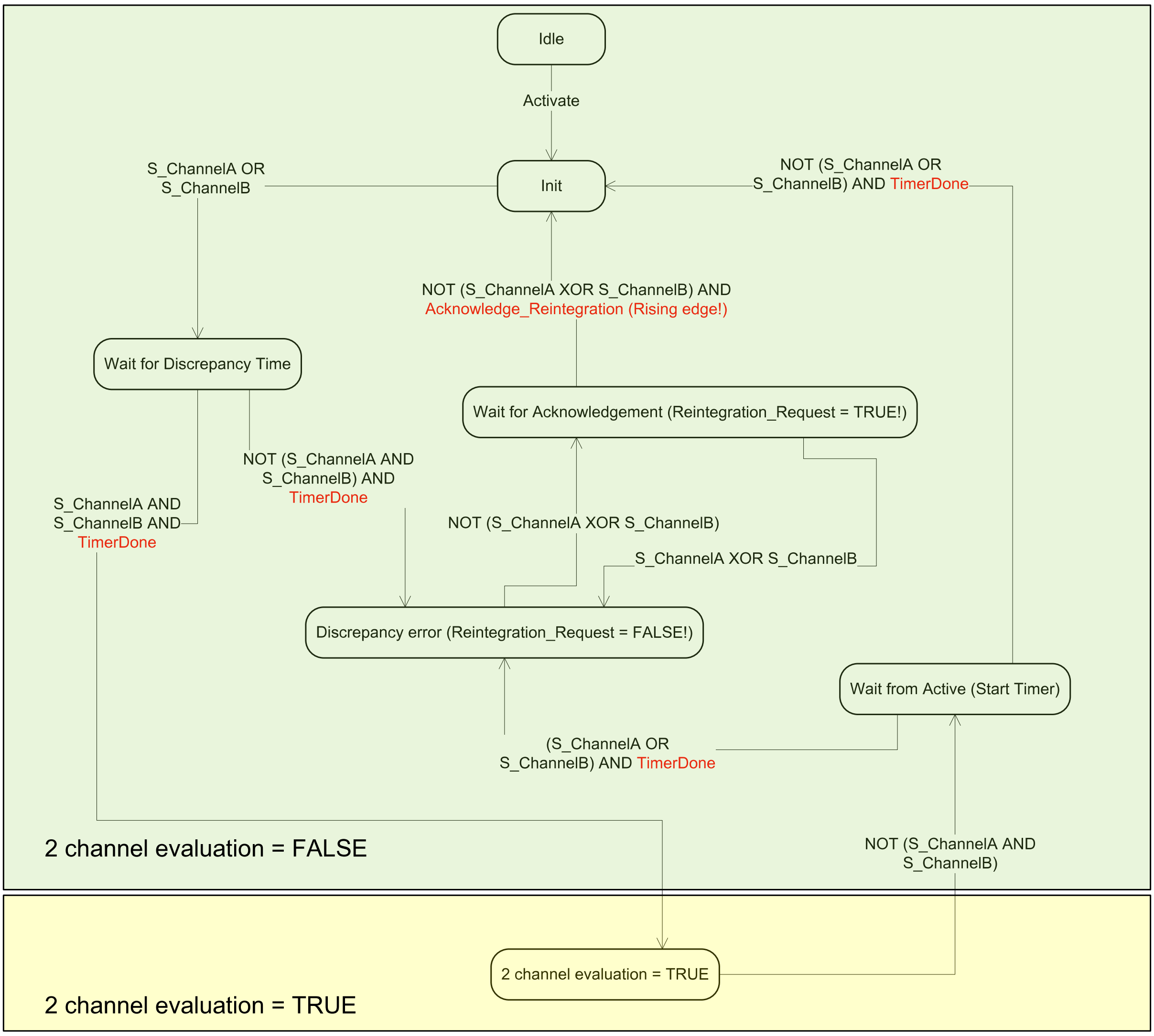


Fig. 38: 2 channel equivalent mode implemented in DX581-S

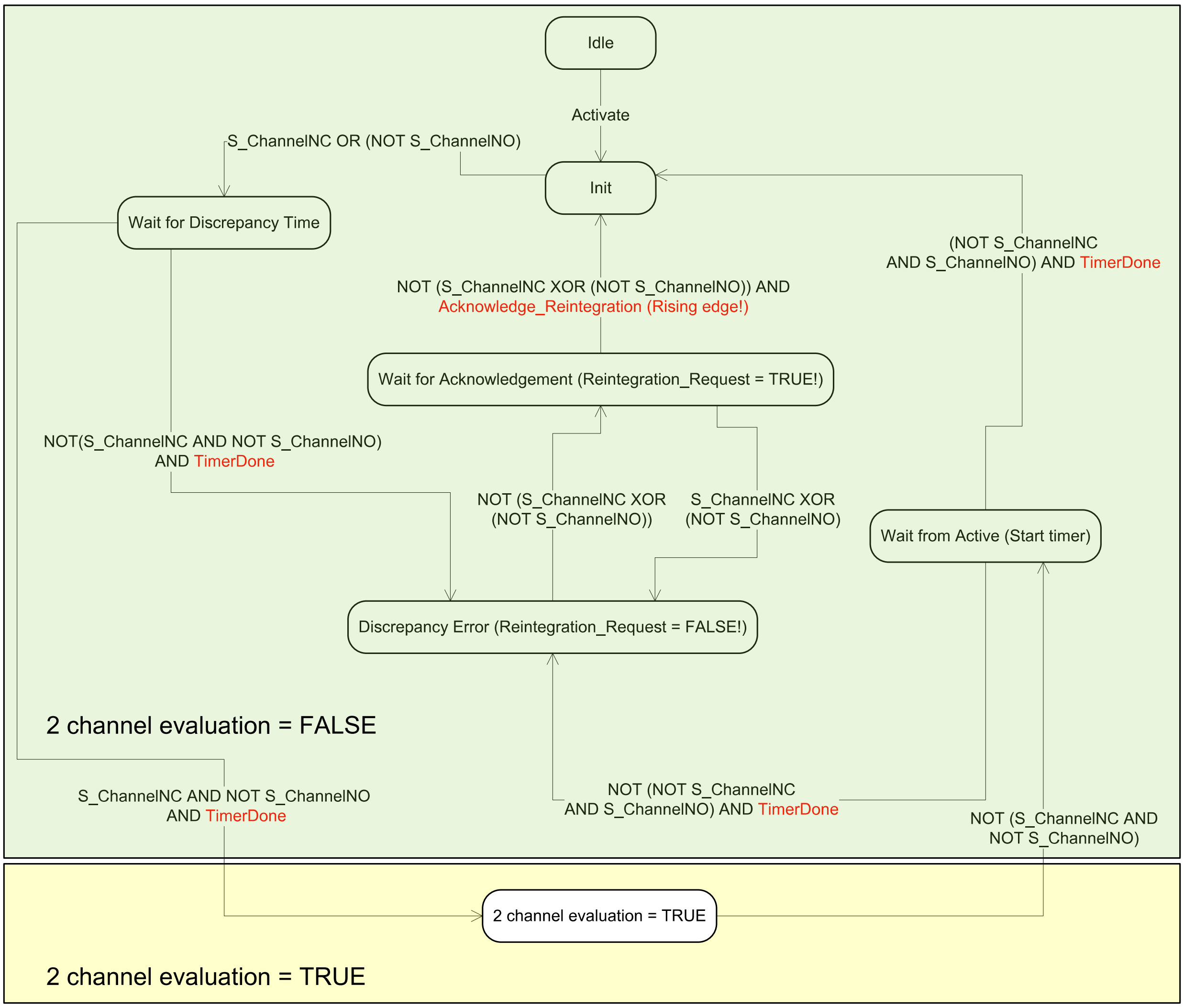


Fig. 39: 2 channel antivalent mode implemented in DX581-S

NOTICE!

2 channel equivalent and 2 channel antivalent modes are implemented in DI581-S and DX581-S module to handle relatively static safety signals, e.g., those for emergency stop devices.

If frequently changing signals, like those from light curtains, laser scanners, door switches, etc. must be handled by DI581-S and DX581-S, then it is highly recommended to use input delay of 1 ms for these channels or to configure related channels in 1 channel mode and do 2 channel equivalent and 2 channel antivalent evaluation at the safety CPU using PLCopen Safety FBs SF_Equivalent *Chapter 4.6.4.2 “SF_Equivalent” on page 202* and SF_Antivalent *Chapter 4.6.4.3 “SF_Antivalent” on page 207*.

DX581-S contains 8 safety digital output channels with the following features:

- Internal output channel tests can be switched off.

DANGER!
If for one of the output channels you set Detection = OFF, the warning appears that the output channel does not satisfy SILCL 3 (IEC 62061) and PL e (ISO 13849-1) requirements in such condition. Two safety output channels may have to be used to satisfy required SILCL or PL level.

The parameter "Detection" was created for customers who want to use safety outputs of DX581-S for SILCL 1 (or maximum SILCL 2 under special conditions) or PL c (or maximum PL d under special conditions) safety functions and have less internal DX581-S pulses visible on the safety output line. Such internal pulses could be detected as LOW signal by, for example, drive inputs, which would lead to unintended machine stop.

DANGER!
Short-circuit to the ground for output channels in DX581-S module is monitored. However, short-circuit to 24 V DC on the output wire is not monitored. End-users have to take appropriate actions (e.g., on the application side by defining appropriate test periods for safety function or by reading back the status of the output wire using one of available safety digital inputs) to satisfy their respective IEC 62061 and ISO 13849-1 requirements, if short-circuit to 24 V DC cannot be excluded.

DANGER!
If an error is detected for the given safety output channel, it is directly passivated by DX581-S module.

Note that for some errors, the reintegration request bit for passivated output channels is automatically set to HIGH as soon as the channel is passivated and the expected LOW state ("0" value) was reached by the output channel. Such behavior can be seen for some errors because DX581-S module is not able in the LOW ("0" value) output channel state to check if previously detected errors which lead to the channel passivation still exist or not.

If the user attempts to reintegrate such output channels using relevant acknowledge reintegration bits, he will succeed but if the error is still present, the relevant channels will be passivated in the next DX581-S error detection cycle.

In the case of internal output module errors, the complete module will be passivated.

3.4.3 Mounting, dimensions and electrical connection

The input/output modules can be plugged only on spring-type TU582-S I/O terminal unit. The unique mechanical coding on I/O terminal units prevents a potential mistake of placing the non-safety I/O module on safety I/O terminal unit and the other way around. Basic information on system assembly is shown here. Detailed information can be found in ⇘ *[4]*.

Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204 part 1, by skilled electricians only.

Assembly of DX581-S

> **DANGER!**
> Hot plug and hot swap of energized modules is not permitted. All power sources (supply and process voltages) must be switched off while working with safety modules.

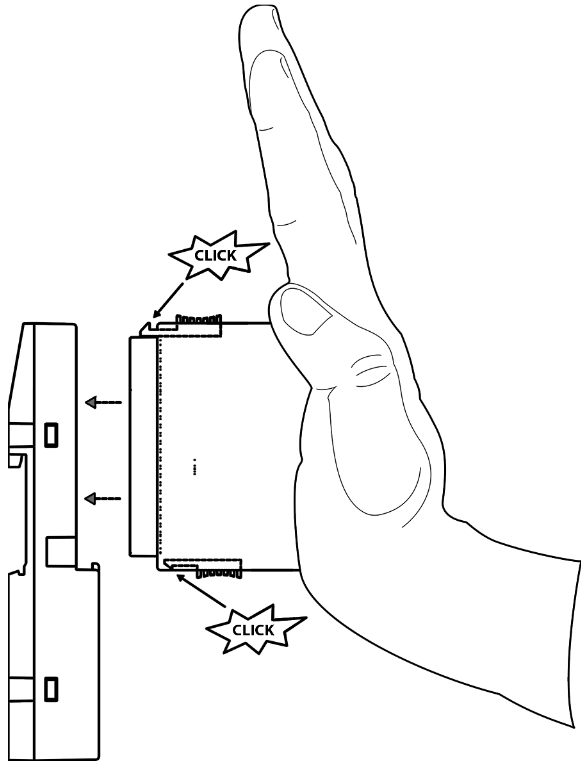


Fig. 40: Assembly instructions

1. Put the module on the terminal unit.
   ⇨ The module clicks in.
2. Then press the module with a force of at least 100 N into the terminal unit to achieve proper electrical contact.

Disassembly of DX581-S

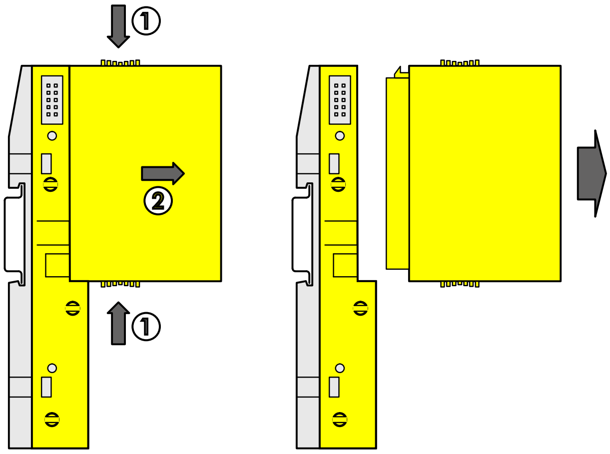


Fig. 41: Disassembly instructions

▷ Press above and below, then remove the module.

Dimensions

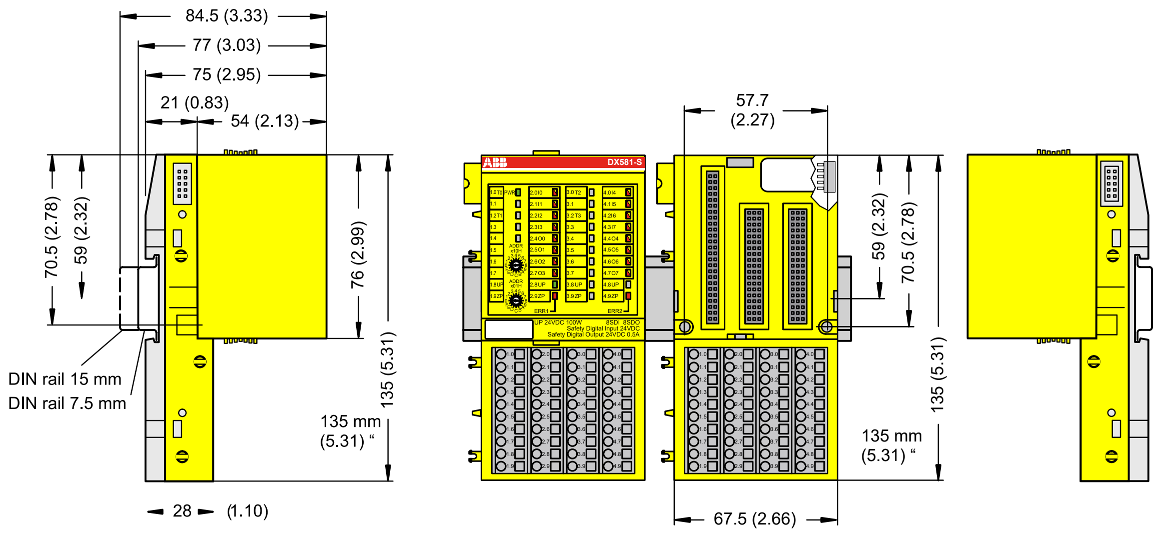


Fig. 42: Dimensions of DX581-S safety I/O module

Electrical connection

> **NOTICE!**
> The same TU582-S is used by all AC500-S safety I/O modules. If TU582-S is wired for DX581-S module with safety digital outputs and DI581-S or AI581-S modules are occasionally placed on this terminal unit, under no circumstances it is possible that safety digital output clamps on TU582-S become energized due to a wrongly placed DI581-S and AI581-S safety I/O modules.

The electrical connection of the I/O channels is carried out using 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8, 2.8, 3.8 and 4.8 as well as 1.9, 2.9, 3.9 and 4.9 are electrically interconnected within the I/O terminal unit and have always the same assignment, independent of the inserted module:

- Terminals 1.8, 2.8, 3.8 and 4.8: Process voltage UP = +24 V DC
- Terminals 1.9, 2.9, 3.9 and 4.9: Process voltage ZP = 0 V

The assignment of the other terminals:

Terminals	Signal	Meaning
1.0, 1.2, 3.0, 3.2	T0, T1, T2, T3	Connectors of 4 test pulse outputs T0, T1, T2, T3
2.0 ... 2.3, 4.0 ... 4.3	I0, I1, I2, I3, I4, I5, I6, I7	8 safety digital inputs
2.4 ... 2.7, 4.4 ... 4.7	O0, O1, O2, O3, O4, O5, O6, O7	8 safety digital outputs
1.8, 2.8, 3.8, 4.8	UP	Process power supply +24 V DC
1.9, 2.9, 3.9, 4.9	ZP	Central process earth
1.1, 1.3, 1.4, 1.5, 1.6, 1.7, 3.1, 3.3, 3.4, 3.5, 3.6, 3.7	Free	Not used

NOTICE!
The process voltage must be included in the earthing concept of the control system (e.g., earthing the minus pole).

Examples of connections

Examples of electrical connections with DX581-S module, single channels Ix and Ox.

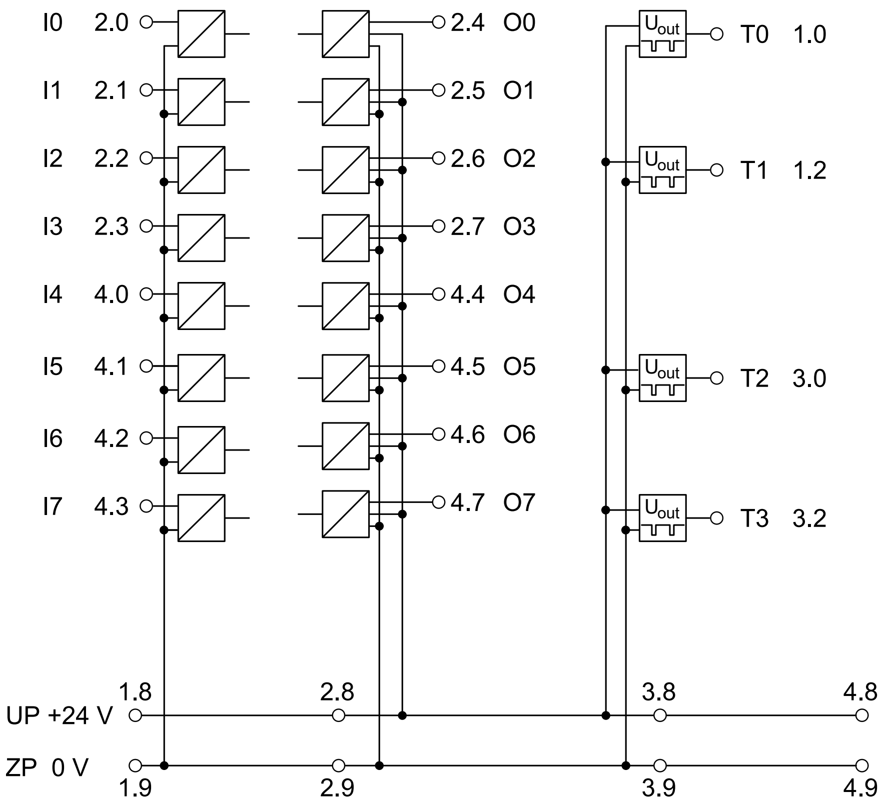


Fig. 43: Example of electrical connections with DX581-S

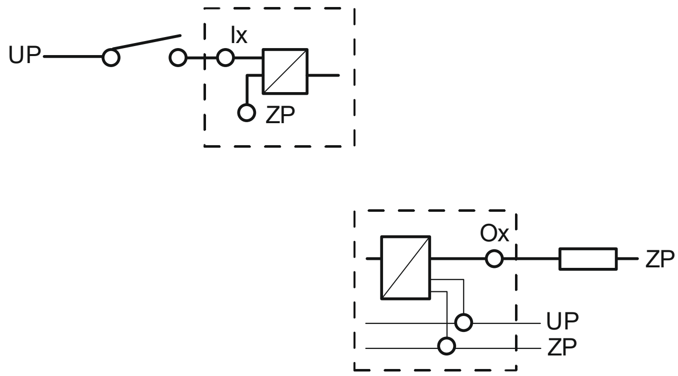


Fig. 44: Example of single channels with DX581-S

3.4.4 Internal data exchange

Inputs (bytes)	5
Outputs (bytes)	3

3.4.5 I/O configuration

The safety digital input/output module DX581-S does not store configuration data itself. The configuration data is stored on the safety and non-safety CPUs.

3.4.6 Parameterization

The arrangement of the parameter data is performed by your system configuration software Automation Builder. ABB GSDML file for PROFINET devices can be used to configure DX581-S parameters in 3rd party PROFINET F-Host systems.

The parameter setting directly influences the functionality of modules and reachable SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1).

No.	Name	Values	Default
1	Check supply	"On", "Off"	"On"
Inputs			
2	Input channel configu-ration	"Not used", "1 channel", "2 channel equivalent", "2 channel antivalent"	"Not used"
3	Test pulse	"Disabled", "Enabled"	"Disabled"
4	Input delay	"1 ms", "2 ms", "5 ms", "10 ms", "15 ms", "30 ms", "50 ms", "100 ms", "200 ms", "500 ms"	"5 ms"
5	Discrepancy time*	"10 ms", "20 ms", "30 ms", "40 ms", "50 ms", "60 ms", "70 ms", "80 ms", "90 ms", "100 ms", "150 ms", "200 ms", "250 ms", "300 ms", "400 ms", "500 ms", "750 ms", "1 s", "2 s", "3 s", "4 s", "5 s", "10 s", "20 s", "30 s"	"50 ms"
Outputs			

No.	Name	Values	Default
6	Output channel configuration	"Not used", "Used"	"Not used"
7	Detection (internal output channel test)	"Off", "On"	"On"

* Available only for "2 channel equivalent" and "2 channel antivalent" configuration

3.4.7 Circuit examples DX581-S

Examples of electrical connections and reachable SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1) with DX581-S module are presented below. Note, that electrical connections presented for DI581-S safety input channels are also valid for DX581-S safety input channels.

NOTICE!

Whenever DC = High is used in the circuit examples with safety digital inputs, the following measure from ISO 13849-1 *[10]* is used with DX581-S module: Cross monitoring of input signals and intermediate results within the logic (L), and temporal and logical software monitor of the program flow and detection of static faults and short circuits (for multiple I/O).

Whenever DC = Medium is used in the circuit examples with safety digital inputs, any of the measures for input devices with DC ≥ 90 % can be used from ISO 13849-1 *[10]*.

NOTICE!

Whenever DC = High is used in the circuit examples with safety digital outputs, the following measure from ISO 13849-1 *[10]* is used with the DX581-S module: Cross monitoring of output signals and intermediate results within the logic (L) and temporal and logical software monitor of the program flow and detection of static faults and short circuits (for multiple I/O).

Whenever DC = Medium is used in the circuit examples with safety digital outputs, any of the measures for output devices with DC ≥ 90 % can be used from ISO 13849-1 *[10]*.

DANGER!

The reachable SILCL (IEC 62061), SIL (IEC 61508) and PL (ISO 13849-1) levels for safety outputs of DX581-S module are only valid if the parameter Detection = "On". If the parameter Detection = "Off" then contact ABB technical support to obtain proper reachable SILCL, SIL and PL levels.

Relay

Internal output channel test	Yes
SILCL / PL [1)]	SILCL 1 / PL c
SIL[2)]	SIL 2
SILCL / PL [3)]	SILCL 2 / PL d
SIL[4)]	SIL 3

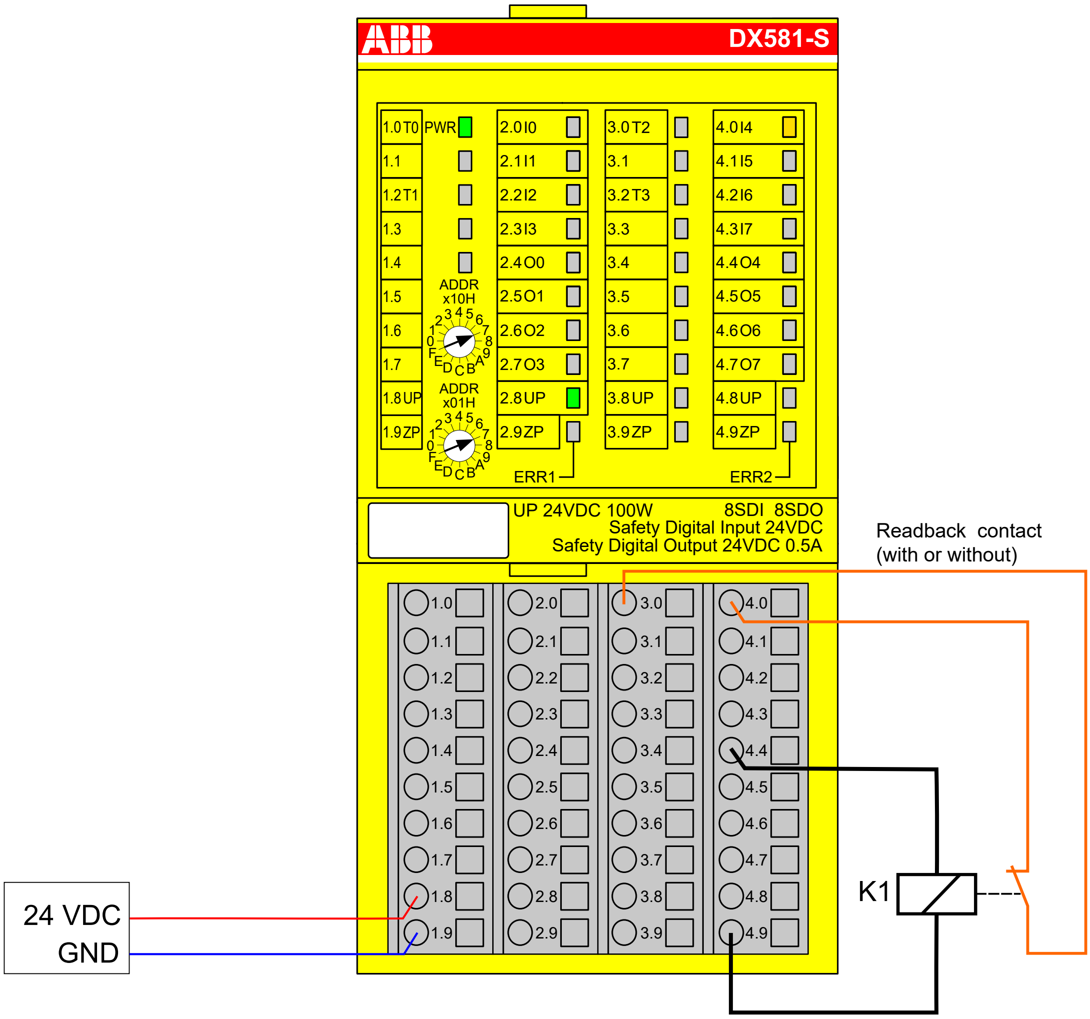


Fig. 45: Circuit example DX581-S, relay

1) - Without readback contact: Max reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion) MTTFd = High; DC = 0

2) - Without readback contact: Max reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher level up to SIL 3 with error exclusion)

3) - With readback contact: Max reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion) MTTFd = High; DC = Medium

4) - With readback contact: Max reachable SIL acc. IEC 61508 (type A components are required)

2 relays

Internal output channel test	Yes
SILCL / PL [1)]	SILCL 1 / PL c
SIL[2)]	SIL 3
SILCL / PL [3)]	SILCL 3 / PL e
SIL[4)]	SIL 3

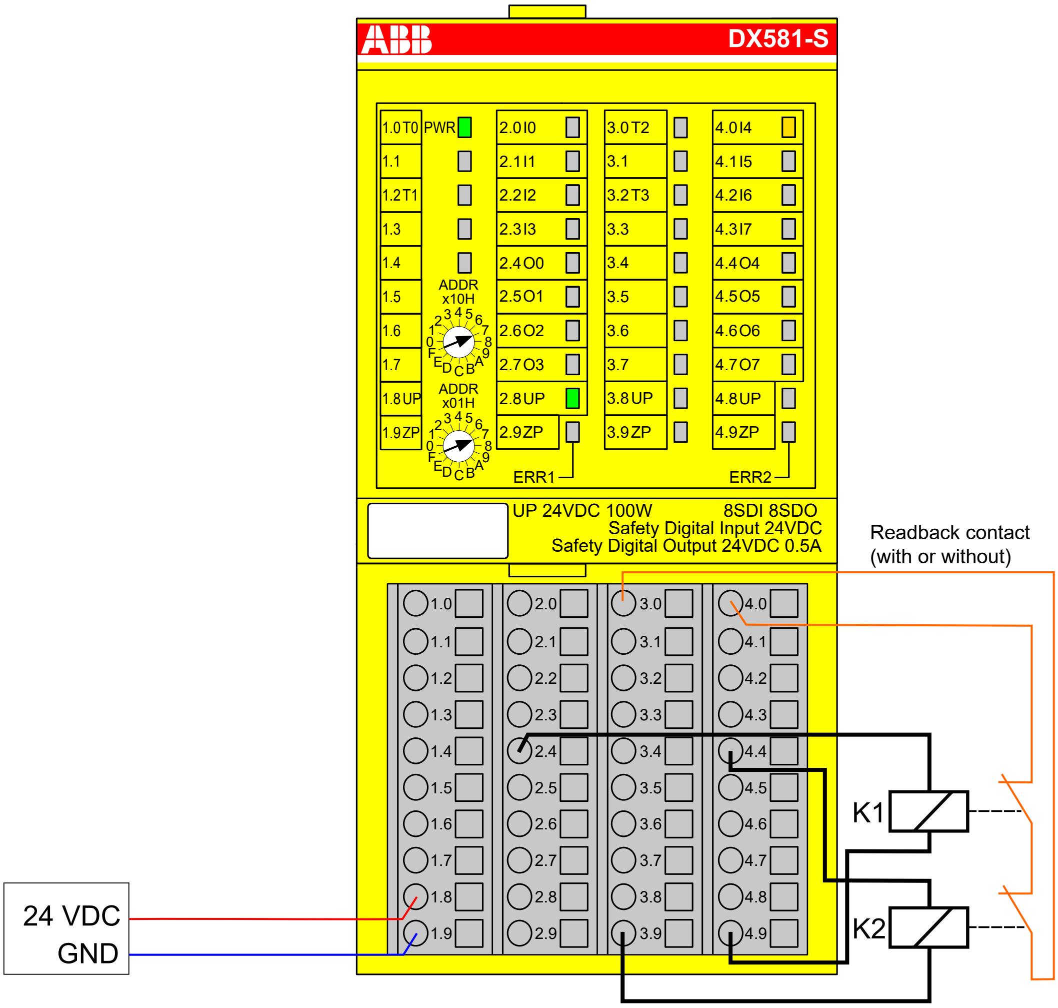


Fig. 46: Circuit example DX581-S, 2 relays

1) - Without readback contact: Max reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion) MTTFd = High; DC = 0

2) - Without readback contact: Max reachable SIL acc. IEC 61508 (type A components are required)

3) - With readback contact: Max reachable (ISO 13849-1, IEC 62061) MTTFd = High; DC = High

4) - With readback contact: Max reachable SIL acc. IEC 61508 (type A components are required)

Device with transistor input (1-channel)

Internal output channel test	Yes
SILCL / PL [1)]	SILCL 1 / PL c
SIL[2)]	SIL 2
SILCL / PL [3)]	SILCL 2 / PL d
SIL[4)]	SIL 3

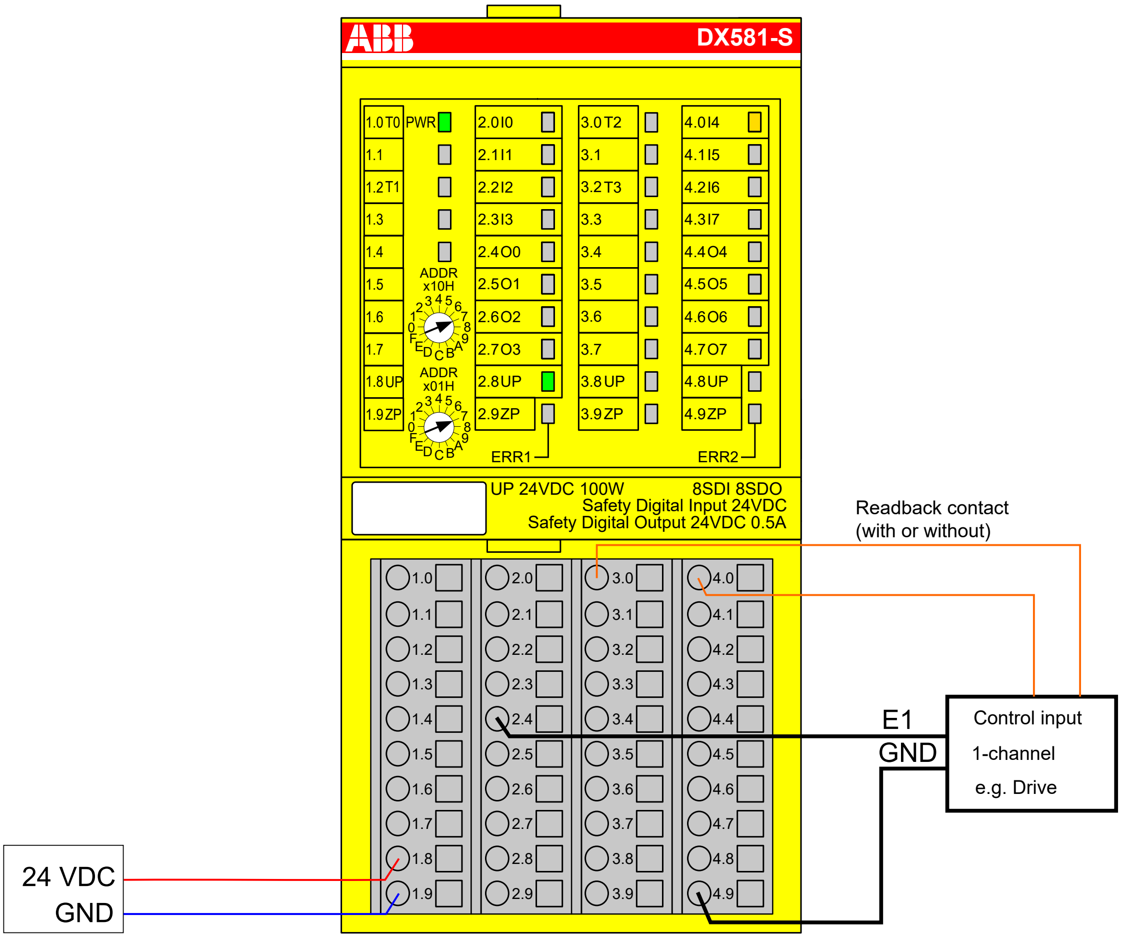


Fig. 47: Circuit example DX581-S, device with transistor input (1-channel)

[1)] - Without readback contact: Max reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion) MTTFd = High; DC = 0

[2)] - Without readback contact: Max reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher level up to SIL 3 with error exclusion)

[3)] - With readback contact: Max reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion) MTTFd = High; DC = Medium

[4)] - With readback contact: Max reachable SIL acc. IEC 61508 (type A components are required)

Device with transistor input (2-channel)

Internal output channel test	Yes
SILCL / PL [1]	SILCL 1 / PL c
SIL[2]	SIL 3
SILCL / PL [3]	SILCL 3 / PL e
SIL[4]	SIL 3

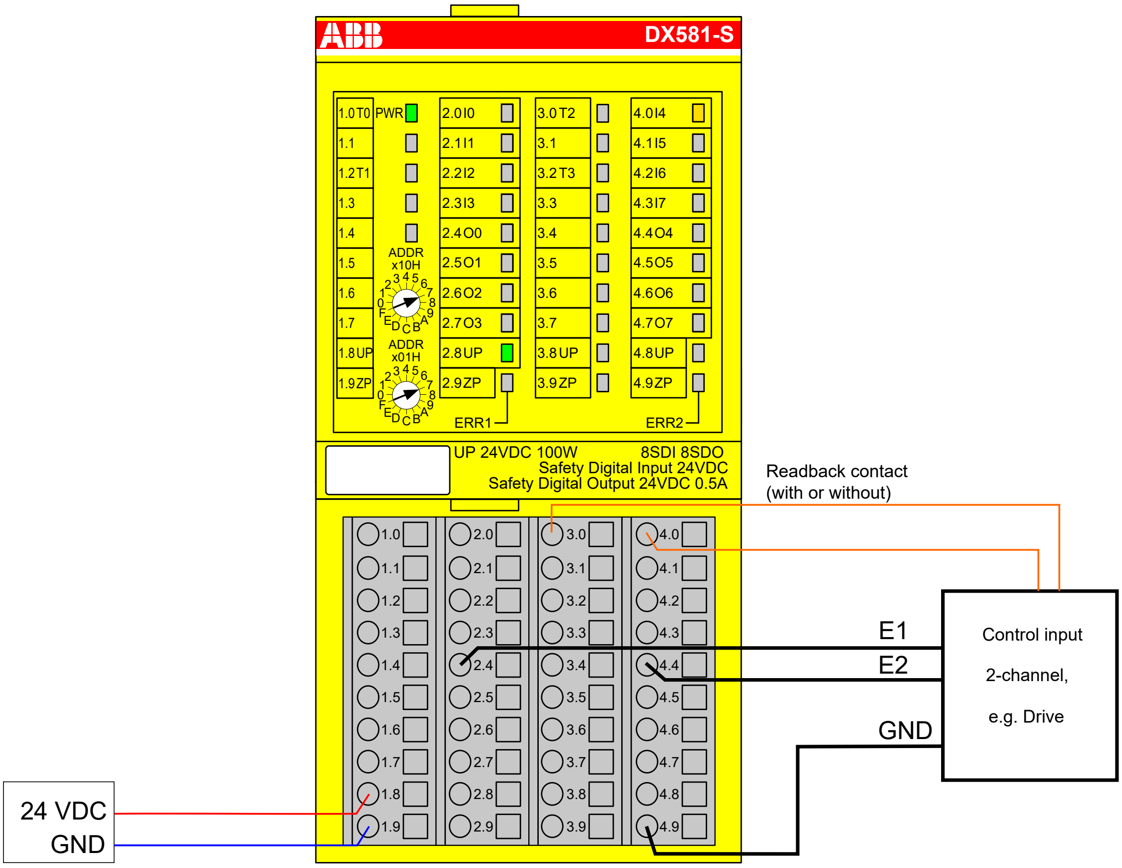


Fig. 48: Circuit example DX581-S, device with transistor input (2-channel)

1) - Without readback contact: Max reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion) MTTFd = High; DC = 0

2) - Without readback contact: Max reachable SIL acc. IEC 61508 (type A components are required)

3) - With readback contact: Max reachable (ISO 13849-1, IEC 62061) MTTFd = High; DC = Medium

4) - With readback contact: Max reachable SIL acc. IEC 61508 (type A components are required)

Error detection on the output wire of lamp, valve, etc.

Internal output channel test	Yes
SILCL / PL [1]	SILCL 2 / PL d Additional dynamic application-specific tests for wiring are required depending on the application and required wiring error detection (short-circuit to 24 V DC, cross-talk error on safety digital outputs, etc.).
SIL[2]	SIL 3

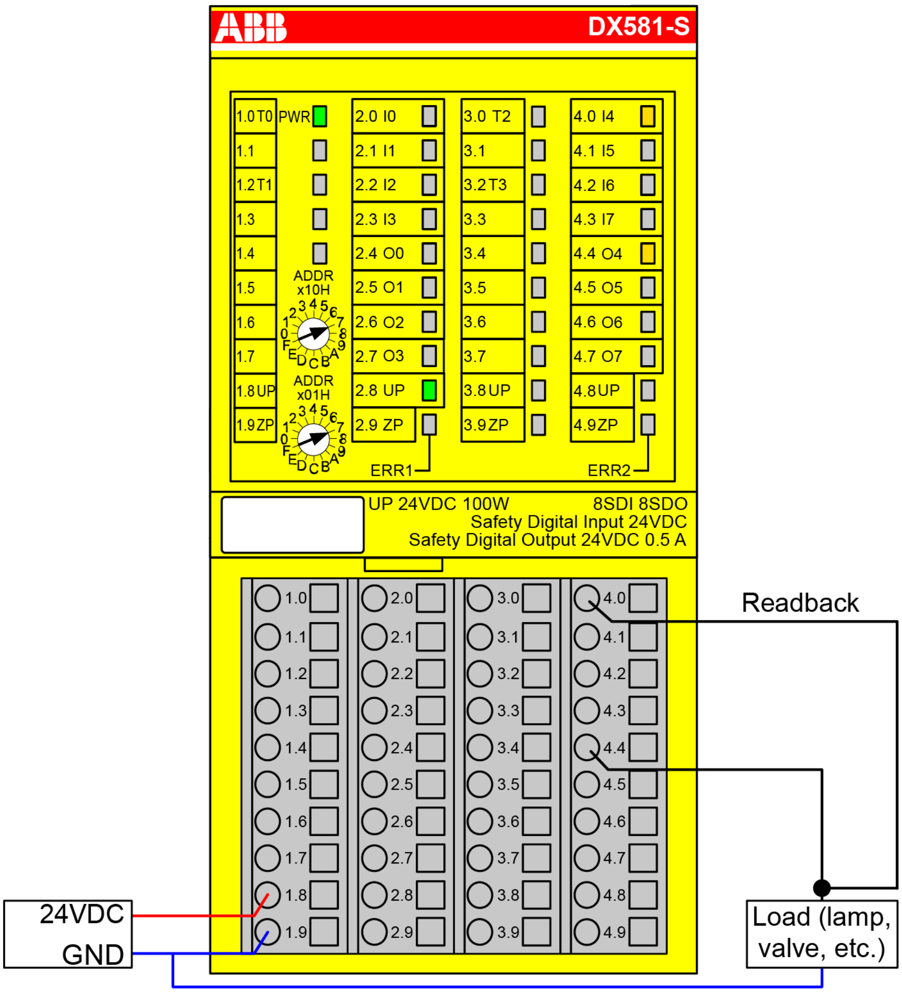


Fig. 49: Circuit example DX581-S, error detection on the output wire of lamp, valve, etc.

1) - Max reachable (ISO 13849, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion) MTTFd = High; DC = Medium

2) - Max reachable SIL acc. IEC 61508 (type A components are required)

Application example

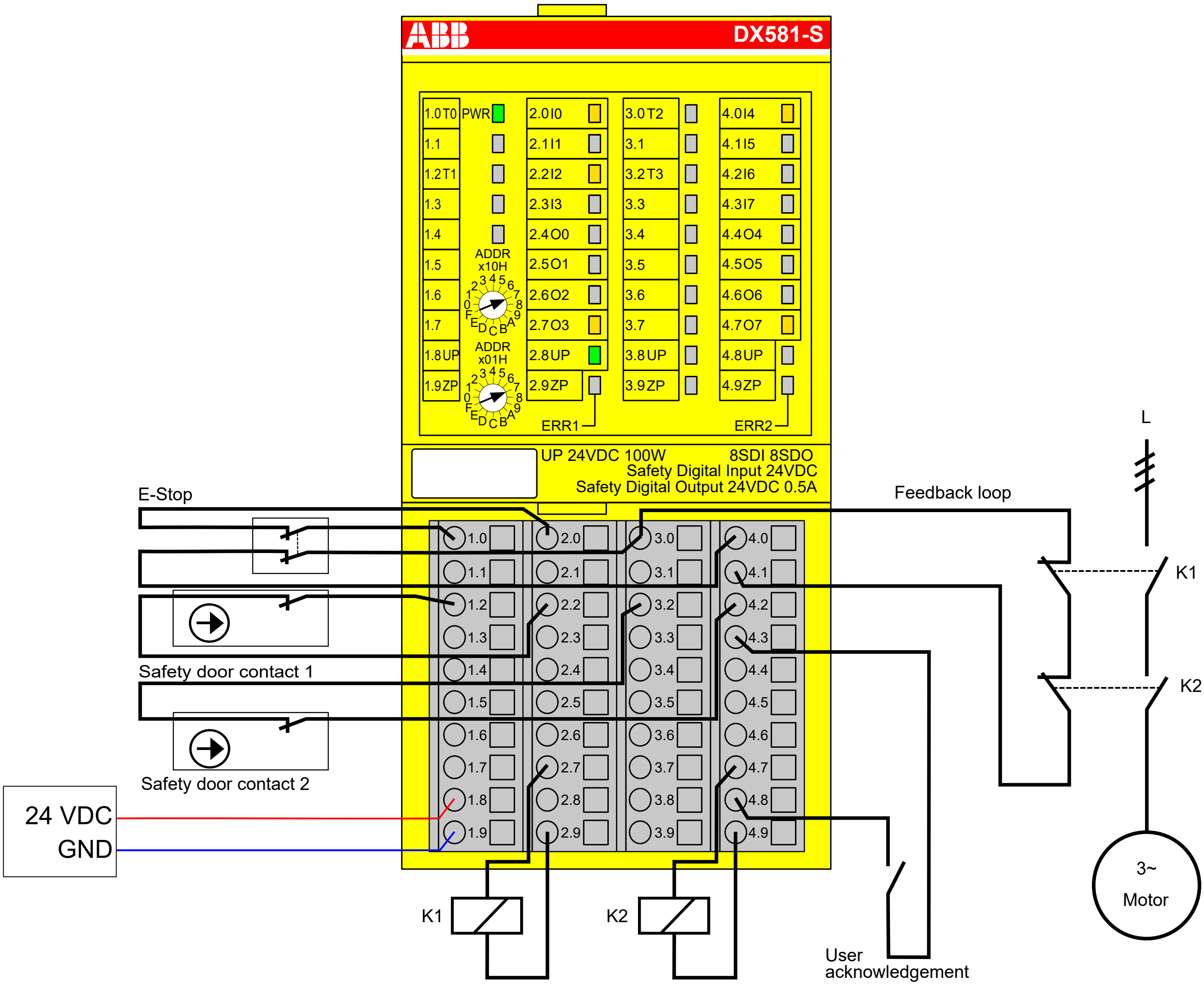


Fig. 50: Application example with DX581-S

3.4.8 LED status display

Table 7: Status display and its meaning

LED	Description	Color	LED = OFF	LED = ON	LED flashes
Inputs 0 ... 7	Digital input	Yellow	Input = OFF	Input = ON (the input voltage is displayed even if the supply voltage is OFF).	-
	Channel error	Red	No channel error	Channel error	-
Outputs 0 ... 7	Digital output	Yellow	Output = OFF	Output = ON	-
	Channel error	Red	No channel error	Channel error	-
UP	Process voltage +24 V DC via terminal	Green	Process supply voltage is missing	Process supply voltage is OK	-

LED	Description	Color	LED = OFF	LED = ON	LED flashes
PWR	+3.3 V voltage from I/O bus	Green	+3.3 V I/O bus voltage is not available	+3.3 V I/O bus voltage is available	-
ERR1	Module error indicator 1	Red	No module error	Module error which leads to a SAFE STOP state	Module passivation and/or acknowledgment request (alternating blinking)
ERR2	Module error indicator 2	Red			

3.4.9 Technical data

NOTICE!
DX581-S-XC version is available for usage in extreme environmental conditions
Appendix A "System data for AC500-S-XC" on page 367.

Additional technical data is available in ABB PLC catalog at *www.abb.com/plc*.

Process supply voltage UP

Data	Value	Unit
Connections terminals 1.8 ... 4.8 (UP)	+24	V
Connections terminals 1.9 ... 4.9 (ZP)	0	V
Rated value (-15 %, +20 %, without ripple)	24	V DC
Max. ripple	5	%
Protection against reversed voltage	yes	
Rated protection fuse for UP (fast)	10	A
Electrical isolation	per module	
Mechanisms in which I/Os are processed	periodically refreshed	
Current consumption from UP at normal operation with + 24 V DC (for module electronics)	0.18	A
Inrush current from UP at 30 V (at power up)	0.1	A^2s
Inrush current from UP at 24 V (at power up)	0.06	A^2s

NOTICE!
All DX581-S channels (including test pulse outputs) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 V DC.

Mounting position

Horizontal or vertical with derating (output load reduced to 50 % at +40 °C per group and with maximal operating temperature reduced to +40 °C).

Cooling

The natural convection cooling must not be hindered by cable ducts or other parts in the switchgear cabinet.

Allowed interruptions of power supply, according to EN 61131-2

Data	Value	Unit
DC supply interruptions	< 10	ms
Time between 2 DC supply interruptions, PS2	> 1	s

Environmental conditions

Data	Value	Unit
Operating temperature*	0 ... +60	°C
Storage temperature	-40 ... +85	°C
Transport temperature	-40 ... +85	°C
Humidity without condensation	max. 95	%
Operating air pressure	> 800	hPa
Storage air pressure	> 660	hPa
Operating altitude	< 2000	m above sea level
Storage altitude	< 3500	m above sea level

* Extended temperature ranges (below 0 °C and above +60 °C) can be supported in special versions of DX581-S *Appendix A “System data for AC500-S-XC” on page 367*.

Creepage distances and clearances

The creepage distances and clearances meet the overvoltage category II, pollution degree 2.

Power supply units

For the supply of modules, power supply units according to PELV/SELV specifications must be used.

Electromagnetic compatibility

For information on electromagnetic compatibility refer to the latest TÜV SÜD Report *[2]*.

Mechanical properties

Data	Value	Unit
Degree of protection	IP 20	
Housing	according to UL 94	
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 3.5 mm	2 ... 15	Hz
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 1 g *	15 ... 150	Hz
Shock test (all three axes), 11 ms half-sinusoidal	15	g
MTBF	73	years

* Higher values on request

Self-test and diagnostic functions

Start-up and runtime tests: Program flow control, RAM, CPU, cross-talk, stuck-at-1, etc.

Dimensions, weight

Data	Value	Unit
W x H x D	67.5 x 76 x 62	mm
Weight	~ 130	g

Certifications CE, cUL (further certifications at www.abb.com/plc)

3.4.9.1 Technical data of safety digital inputs

Data	Value	Unit
Number of input channels per module	8	
Terminals of the channels I0 to I3	2.0 ... 2.3	
Terminals of the channels I4 to I7	4.0 ... 4.3	
Terminals of reference potential for all inputs (minus pole of the process supply voltage, signal name ZP)	1.9 ... 4.9	
Electrical isolation from the rest of the module (I/O bus)	Yes	
Input type acc. to EN 61131-2	Type 1	
Input delay (0 ➔ 1 or 1 ➔ 0), configurable	1 ... 500	ms

Input signal indication

One yellow LED per channel, the LED is ON when the input signal is high (signal 1).

Signal voltage

Data	Value	Unit
Input signal voltage	24	V DC
Signal 0	-3 ... +5	V
Undefined signal	> +5 ... < +15	V
Signal 1	+15 ... +30	V

Input current per channel

Data	Value	Unit
Input voltage +24 V, typically	7	mA
Input voltage +5 V	> 1	mA
Input voltage +15 V	> 4	mA
Input voltage +30 V	< 8	mA

Cable length

Data	Value	Unit
Max. cable length, shielded	1000	m
Max. cable length, unshielded	600	m

3.4.9.2 Technical data of safety digital outputs

DANGER!
Exceeding the permitted process or supply voltage range (< -35 V DC or > +35 V DC) could lead to unrecoverable damage of the system.

Data	Value	Unit
Number of channels per module (transistor outputs)	8	
Terminals of reference potential for all outputs (minus pole of the process supply voltage, signal name ZP)	1.9 ... 4.9	
Terminals of common power supply voltage for all outputs (plus pole of the process supply voltage, signal name UP)	1.8 ... 4.8	
Output voltage for signal 1	UP - 3	V
Output delay (0 ➔ 1 or 1 ➔ 0): 5 mA output current	1	ms
Output delay (0 ➔ 1 or 1 ➔ 0): 500 mA output current	4	ms
Ability to switch a capacitive load of at least	300	μF
Ability to switch an inductive load of at least	1	H

Output current

Data	Value	Unit
Rated value, per channel at UP = 24 V	500	mA
Maximum value (all channels together)	4	A
Leakage current with signal 0	< 0.5	mA
Short-circuit proof/overload proof	yes	
Overload message (channel passivation), I > 0.7 A	yes	
Output current limitation (automatic reactivation after short-circuit/overload)	yes	
Resistance to feedback against 24 V signals	yes	
Demagnetization by internal suppressor diodes when switching off inductive loads	yes	
Rated protection fuse on UP	4.5	A

Cable length

Data	Value	Unit
Max. cable length, shielded	1000	m
Max. cable length, unshielded	600	m

3.4.9.3 Technical data of non-safety test pulse outputs

Data	Value	Unit
Number of test pulse channels per module (transistor test pulse outputs)	4	
Terminals of the channels T0, T1	1.0, 1.2	
Terminals of the channels T2, T3	3.0, 3.2	
Terminals of reference potential for all test pulse outputs (minus pole of the process supply voltage, signal name ZP)	1.9 ... 4.9	
Terminals of common power supply voltage for all outputs (plus pole of the process supply voltage, signal name UP)	1.8 ... 4.8	
Output voltage for signal 1	UP - 0.8	V

Data	Value	Unit
Length of test pulse 0 phase	1	ms

Output current

Data	Value	Unit
Rated value, per channel	10	mA
Maximum value (all channels together)	40	mA
Short-circuit proof / overload proof	yes	
Output current limitation	65	mA
Resistance to feedback against 24 V signals	yes	

Cable length

Data	Value	Unit
Max. cable length, shielded	1000	m
Max. cable length, unshielded	600	m

3.4.10 Ordering data

Type	Description	Part no.
DX581-S	Safety digital I/O module 8SDI/SDO	1SAP 284 100 R0001
DX581-S-XC	Safety digital I/O module 8SDI/SDO, extreme conditions	1SAP 484 100 R0001

3.5 AI581-S safety analog input module

Elements of the module

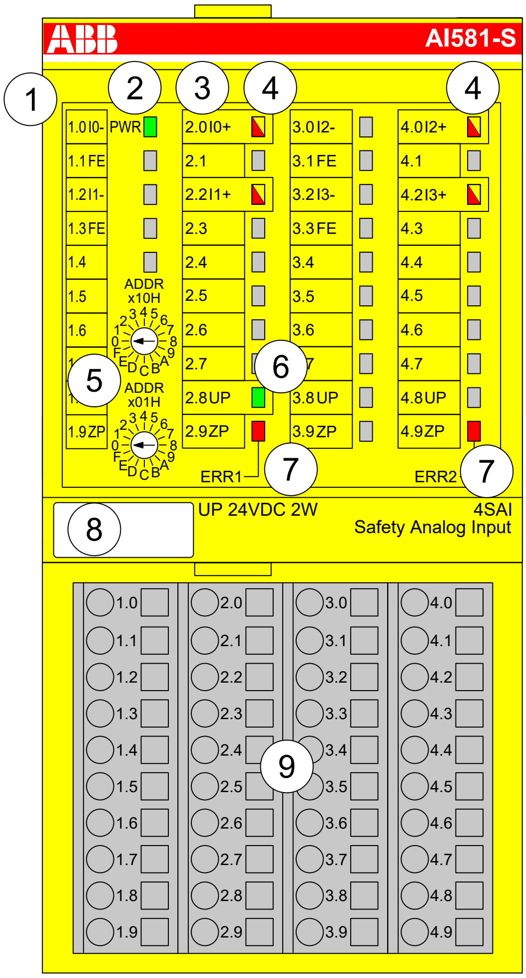


Fig. 51: Safety analog input module AI581-S, plugged on terminal unit TU582-S

1 I/O bus
2 System LED
3 Allocation terminal no. - signal name
4 4 yellow/red LEDs signal status I0 ... I1/I2 ... I3
5 2 rotary switches for PROFIsafe address
6 Green LED for process voltage UP
7 Red LEDs to display module errors
8 Label (TA525)
9 I/O terminal unit (TU582-S)

3.5.1 Purpose

Safety analog input module AI581-S can be used as a remote expansion module at CI501-PNIO, CI502-PNIO, CI504-PNIO and CI506-PNIO PROFINET modules or locally at AC500 CPUs for up to SIL 3 (IEC 61508), SILCL 3 (IEC 62061) and PL e (ISO 13849-1) safety applications.

> **NOTICE!**
> SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1) reachable in your safety application depend on the wiring of your sensors to AI581-S module *Chapter 3.5.7 “Circuit examples AI581-S” on page 117*.

AI581-S contains 4 safety current analog inputs separated in two groups (2.0 ... 2.2 and 4.0 ... 4.2) with no potential separation between the channels.

The inputs are not electrically isolated from the other electronic circuitry of the module.

3.5.2 Functionality

Analog inputs	4 (0 ... 20 mA or 4 ... 20 mA)
LED displays	for signal status, module errors, channel errors and supply voltage
Internal power supply	through the expansion bus interface (I/O bus)
External power supply	via the terminals ZP and UP (process voltage 24 V DC)

Self-tests and diagnostic functions (both start-up and runtime), like CPU and RAM tests, program flow control and cross-talk tests, etc. are implemented in AI581-S according to IEC 61508 SIL 3 requirements.

> **NOTICE!**
> Only F_Dest_Add is used for PROFIsafe F-Device identification in AI581-S.

AI581-S contains 4 safety analog input channels with the following features:

- 14 bit resolution.
- Checking of process power supply (diagnostic message is sent from the safety I/O module to the CPU informing about the lack of process power supply for the given safety I/O module). This function is a non-safety one and is not related to the internal safety-relevant over- and undervoltage detection.
- Noise rejection 50 Hz or 60 Hz.
- 1 channel (0 ... 20 mA), 1 channel (4 ... 20 mA) or 2 channel (4 ... 20 mA) modes (minimum or maximum value can be selected for transfer to safety CPU in 2 channel (4 ... 20 mA) mode; tolerance range 4 ... 12 % can be set for 2 channel mode).

> **NOTICE!**
> In a 2 channel mode, the lower channel (channels 0/2 ➔ Channel 0, channels 1/3 ➔ Channel 1, etc.) transports the aggregated process value, PROFIsafe diagnostic bit, acknowledgment request and acknowledge reintegration information. The higher channel always provides the passivated value "0".

> **NOTICE!**
> The maximal internal discrepancy time between two internal channel values (1 channel or 2 channel modes) in AI581-S module is 67.5 ms, which is also an internal worst-case input delay value.
>
> The discrepancy time between two channel values (2 channel mode) with the selected supervised tolerance range (4 ... 12 %) is also 67.5 ms.

> **NOTICE!**
> The analog input channels have built-in hardware low-pass filter of 100 Hz.

> **NOTICE!**
> In case of the overcurrent/undercurrent detected at the safety analog input channel, the channel passivation takes place latest after 200 ms. The channel remains passivated for 30 s and then the check is performed if the overcurrent/undercurrent still present or not. If the overcurrent/undercurrent has gone, then reintegration request signal for the given channel is set to TRUE to allow channel reintegration.

The following table shows the mapping of safety CPU process values to the values in mA from AI581-S module. Two modes are defined for an analog input 0 ... 20 mA and 4 ... 20 mA.

> **NOTICE!**
> Both overflow and overrange represent an overcurrent. Both underflow and underrange represent an undercurrent.
>
> Only in case of overflow and underflow, the analog channels are passivated and "0" process values are delivered to the safety CPU.

Range	0 ... 20 mA	4 ... 20 mA	Digital value (dec)		Digital value (hex)	
Overflow*	: > 23.519	: > 22.81	32767* : 32512*		7FFF* : 7F00*	
Overrange	23.519 : 20.000723	22.81 : 20.000578	32511 : 27649		7EFF : 6C01	
Nominal range	20 : 0	20 : 16 : 4	27648 : 20736 : 0		6C00 : 5100 : 0000	
Underrange	-0.000723 : -1.481	3.999421 : 1.185	**0 ... 20 mA** -1 : -2048	**4 ... 20 mA** -1 : -4864	**0 ... 20 mA** FFFF : F800	**4 ... 20 mA** FFFF : ED00
Underflow*	< -1.481	< 1.185	**0 ... 20 mA** -2049* : -32768*	**4 ... 20 mA** -4865* : -32678*	**0 ... 20 mA** F7FF* : 8000*	**4 ... 20 mA** ECFF* : 8000*

* In these cases, the analog channels are passivated and "0" process values are delivered to the safety CPU.

3.5.3 Mounting, dimensions and electrical connection

The input modules can be plugged only on spring-type TU582-S I/O terminal unit. The unique mechanical coding on I/O terminal units prevents a potential mistake of placing the non-safety I/O module on safety I/O terminal unit and the other way around. Basic information on system assembly is shown here. Detailed information can be found in ↬ *[4]*.

Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204 part 1, by skilled electricians only.

Assembly of AI581-S

> **DANGER!**
> Hot plug and hot swap of energized modules is not permitted. All power sources (supply and process voltages) must be switched off while working with safety modules.

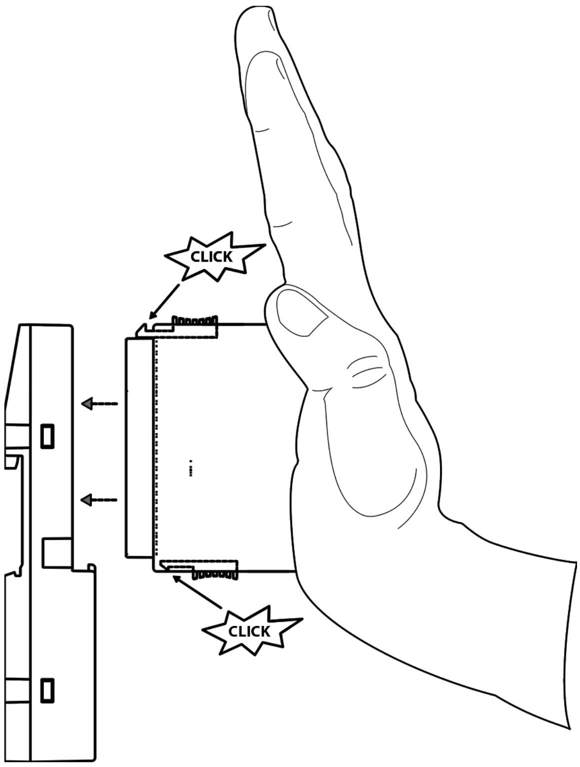


Fig. 52: Assembly instructions

1. Put the module on the terminal unit.

   ⇨ The module clicks in.

2. Then press the module with a force of at least 100 N into the terminal unit to achieve proper electrical contact.

Disassembly of AI581-S

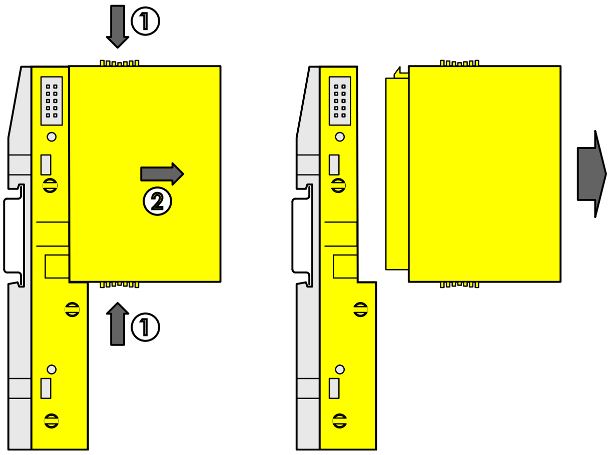


Fig. 53: Disassembly instructions

▷ Press above and below, then remove the module.

Dimensions

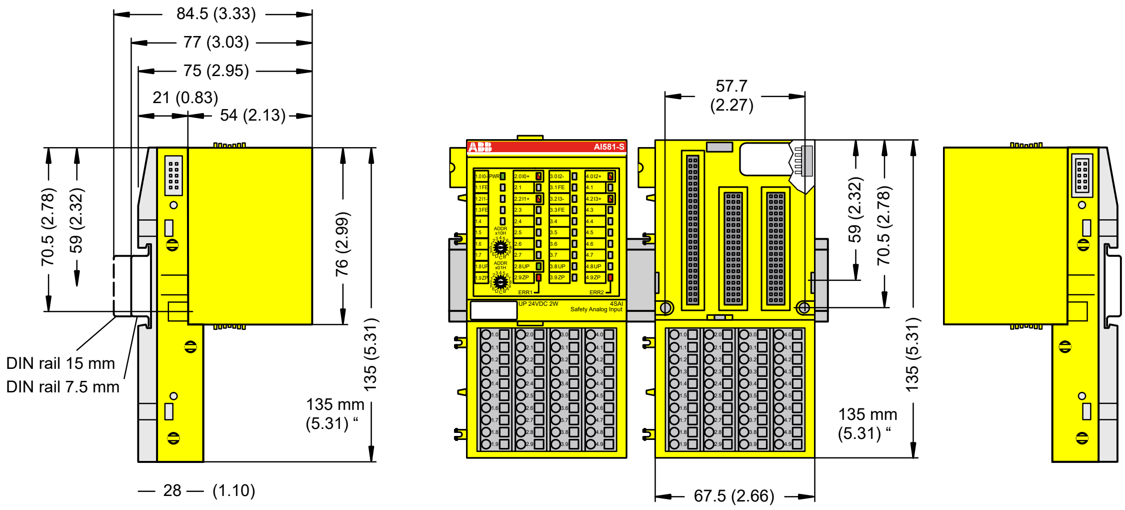


Fig. 54: Dimensions of AI581-S safety I/O module

Electrical connection

> **NOTICE!**
> The same TU582-S is used by all AC500-S safety I/O modules. If TU582-S is wired for DX581-S module with safety digital outputs and DI581-S or AI581-S modules are occasionally placed on this terminal unit, under no circumstances it is possible that safety digital output clamps on TU582-S become energized due to a wrongly placed DI581-S and AI581-S safety I/O modules.

The electrical connection of the I/O channels is carried out using 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8, 2.8, 3.8 and 4.8 as well as 1.9, 2.9, 3.9 and 4.9 are electrically interconnected within the I/O terminal unit and have always the same assignment, independent of the inserted module:

- Terminals 1.8, 2.8, 3.8 and 4.8: Process voltage UP = +24 V DC
- Terminals 1.9, 2.9, 3.9 and 4.9: Process voltage ZP = 0 V

The assignment of the other terminals:

Terminals	Signal	Meaning
1.0, 1.2, 3.0, 3.2	I0-, I1-, I2-, I3-	Negative connectors of 4 analog inputs
2.0, 2.2, 4.0, 4.2	I0+, I1+, I2+, I3+	Positive connectors of 4 analog inputs
1.1, 1.3, 3.1, 3.3	FE	Functional earth
1.8, 2.8, 3.8, 4.8	UP	Process power supply +24 V DC
1.9, 2.9, 3.9, 4.9	ZP	Central process earth
1.4 ... 1.7, 2.1, 2.3 ... 2.7, 3.4 ... 3.7, 4.1, 4.3 ... 4.7	Free	Not used

NOTICE!
The process voltage must be included in the earthing concept of the control system (e.g., earthing the minus pole).

NOTICE!
The minus poles of the analog inputs are electrically connected to each other. They form an "analog ground" signal for the module.

Because of their common reference potential, analog current inputs cannot be circuited in series, neither within the module nor with channels of other modules.

NOTICE!
There is no electrical isolation between the analog circuitry and ZP/UP. Therefore, analog sensors must be electrically isolated in order to avoid loops via the earth potential or supply voltage.

NOTICE!
Analog signals are always laid in shielded cables. The cable shields are earthed at both ends of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

For simple applications (low disturbances, no high requirement on precision), the shielding can also be omitted.

Examples of connections

Examples of electrical connections with AI581-S module and single channels Ix.

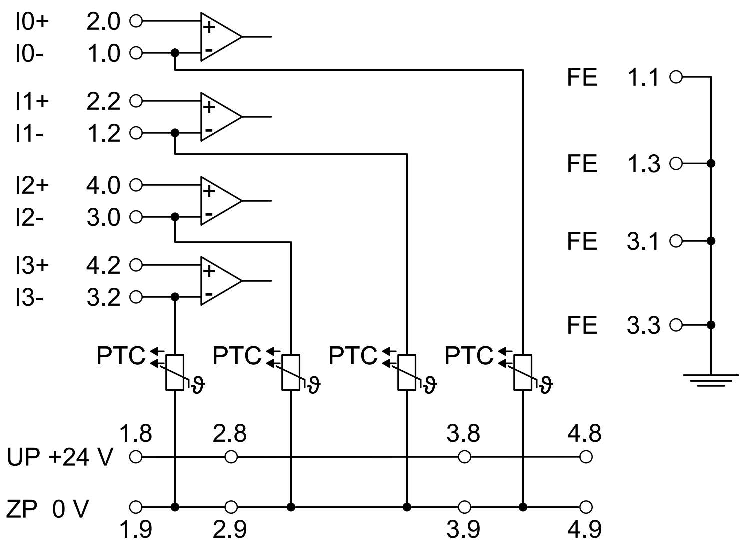


Fig. 55: Example of electrical connections with AI581-S

> **NOTICE!**
> The PTC shown in the connection diagram is built-in in AI581-S module.

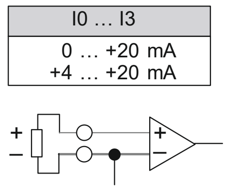


Fig. 56: Example of single channels with AI581-S

3.5.4 Internal data exchange

Inputs (bytes)	9
Outputs (bytes)	1

3.5.5 I/O configuration

The safety analog input module AI581-S does not store configuration data itself. The configuration data is stored on the safety and non-safety CPUs.

3.5.6 Parameterization

The arrangement of the parameter data is performed by your system configuration software Automation Builder. ABB GSDML file for PROFINET devices can be used to configure AI581-S parameters in 3rd party PROFINET F-Host systems.

The parameter setting directly influences the functionality of modules and reachable SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1).

No.	Name	Values	Default
1	Check supply	"On", "Off"	"On"
2	Configuration	"Not used", "1 channel (0 ... 20 mA)", "1 channel (4 ... 20 mA)", "2 channel (4 ... 20 mA)"	"Not used"
3	Noise rejection	"50 Hz", "60 Hz", "None"	"50 Hz"
4	Tolerance range (used only for "2 channel (4 ... 20 mA)" mode)	"4 %", "5 %", "6 %", "7 %", "8 %", "9 %", "10 %", "11 %", "12 %"	"4 %"
5	Used value (min/max) (used only for "2 channel (4 ... 20 mA)" mode)	"Minimum", "Maximum"	"Minimum"

3.5.7 Circuit examples AI581-S

Examples of electrical connections and reachable SIL (IEC 61508), SILCL (IEC 62061) and PL (ISO 13849-1) with AI581-S module are presented below.

NOTICE!

Whenever DC = High is used in the circuit examples with safety analog inputs, the following measure from ISO 13849-1 *[10]* is used with AI581-S module: Cross monitoring of input signals and intermediate results within the logic (L), and temporal and logical software monitor of the program flow and detection of static faults and short circuits (for multiple I/O).

Whenever DC = Medium is used in the circuit examples with safety analog inputs, any of the measures for input devices with DC ≥ 90 % can be used from ISO 13849-1 *[10]*.

Analog sensor (0 ... 20 mA)

SILCL / PL 1), 2)	SILCL 1 / PL c
SIL 3)	SIL 1

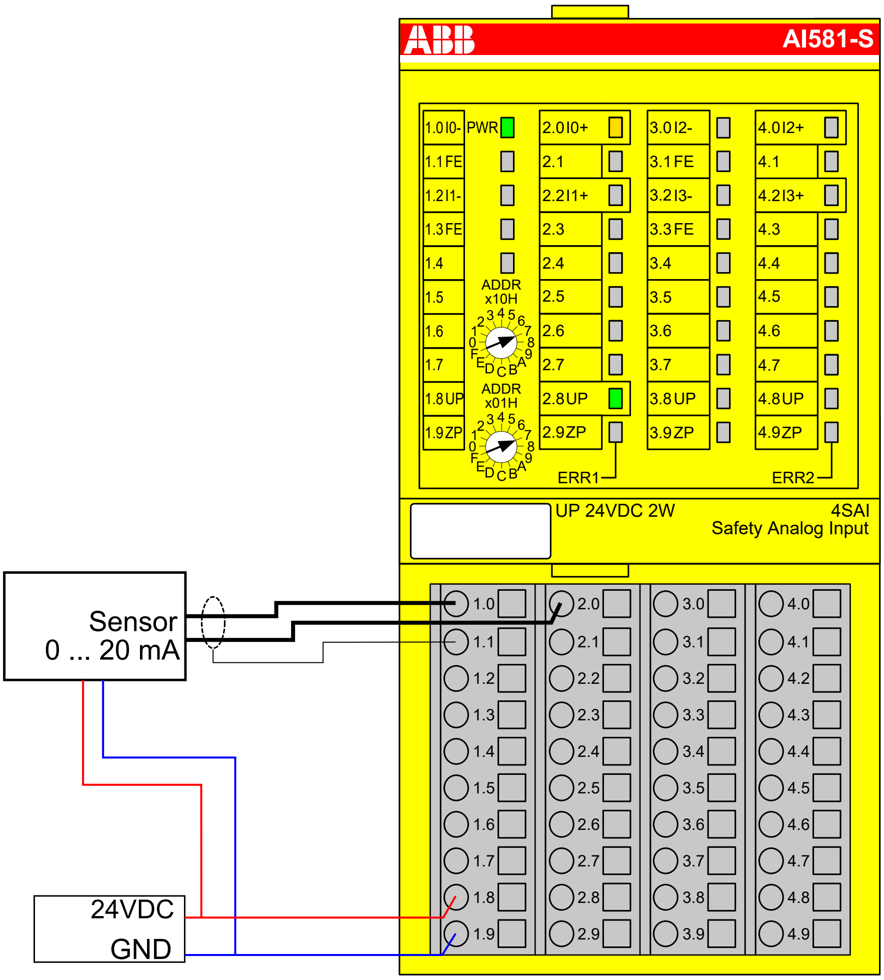


Fig. 57: Circuit example AI581-S, analog sensor (0 ... 20 mA)

1) - MTTFd = High, DC = Low

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher levels up to SIL 3 with error exclusion)

2 analog sensors (0 ... 20 mA)

2-channel evaluation	In AI581-S module
SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 3

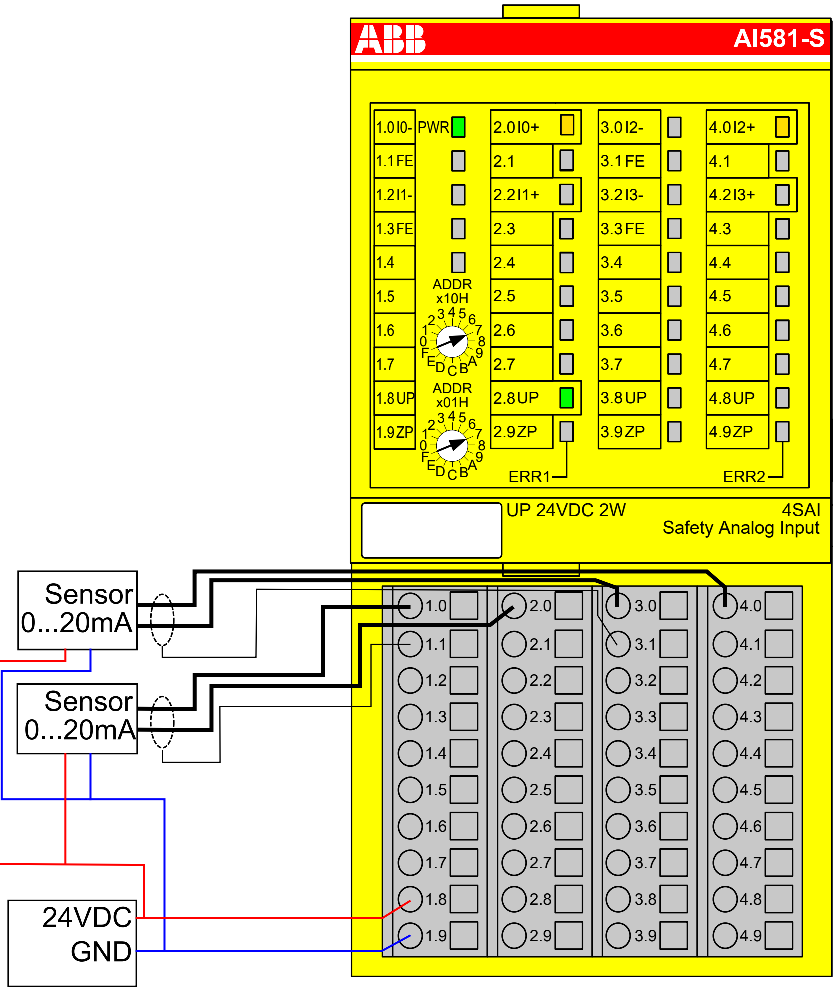


Fig. 58: Circuit example AI581-S, 2 analog sensors (0 ... 20 mA)

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

Analog sensor (4 ... 20 mA)

SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 2

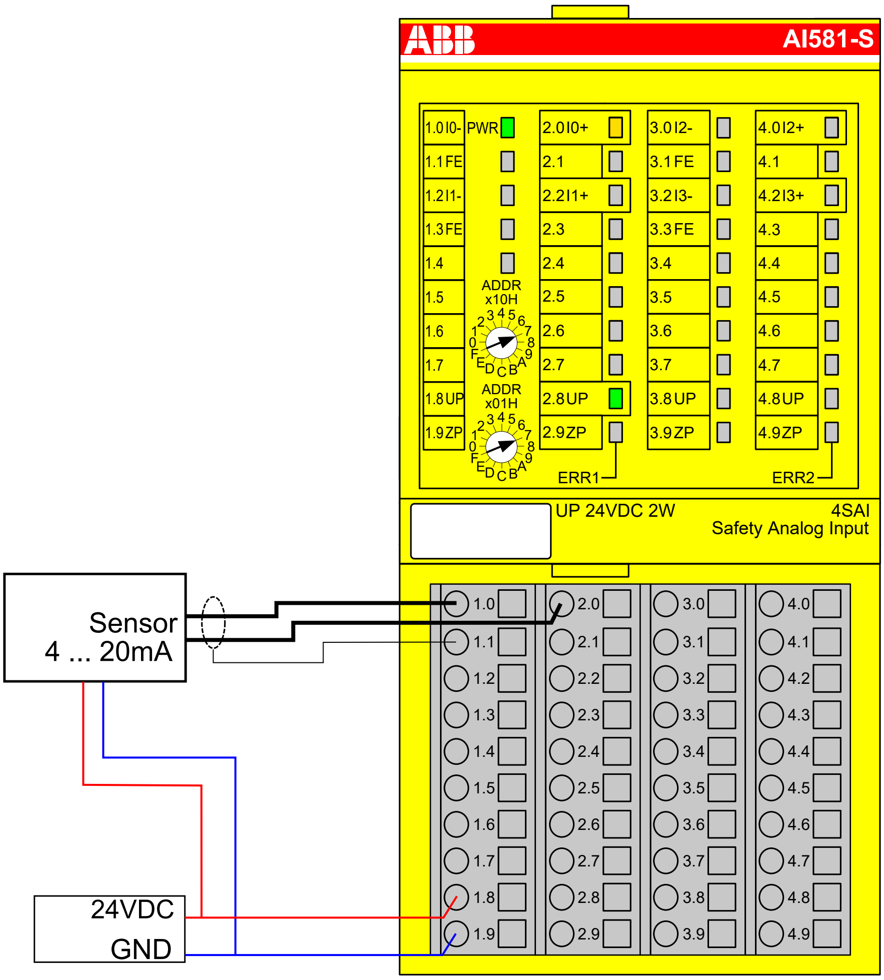


Fig. 59: Circuit example AI581-S, analog sensor (4 ... 20 mA)

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher levels up to SIL 3 with error exclusion)

2 analog sensors (4 ... 20 mA)

2-channel evaluation	In AI581-S module
SILCL / PL 1), 2)	SILCL 3 / PL e
SIL3)	SIL 3

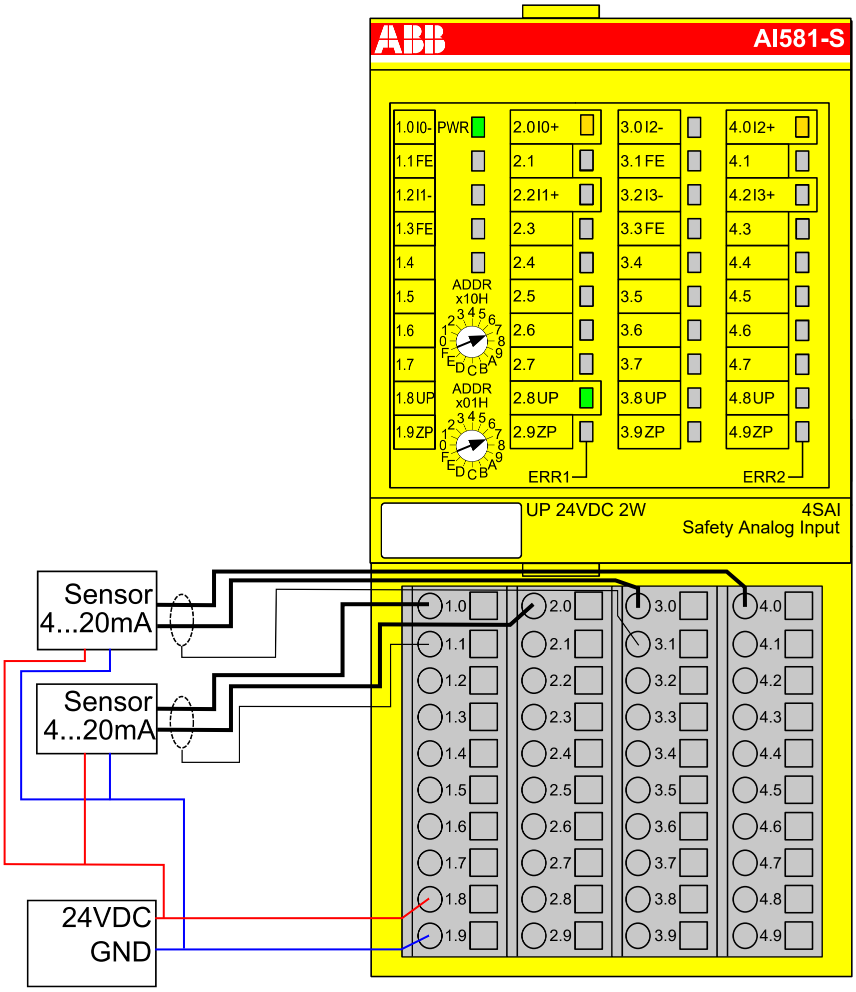


Fig. 60: Circuit example AI581-S, 2 analog sensors (4 ... 20 mA)

1) - MTTFd = High, DC = High

2) - Max. reachable (ISO 13849-1, IEC 62061)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

Transmitter (4 ... 20 mA)

SILCL / PL [1), 2)]	SILCL 2 / PL d
SIL[3)]	SIL 2

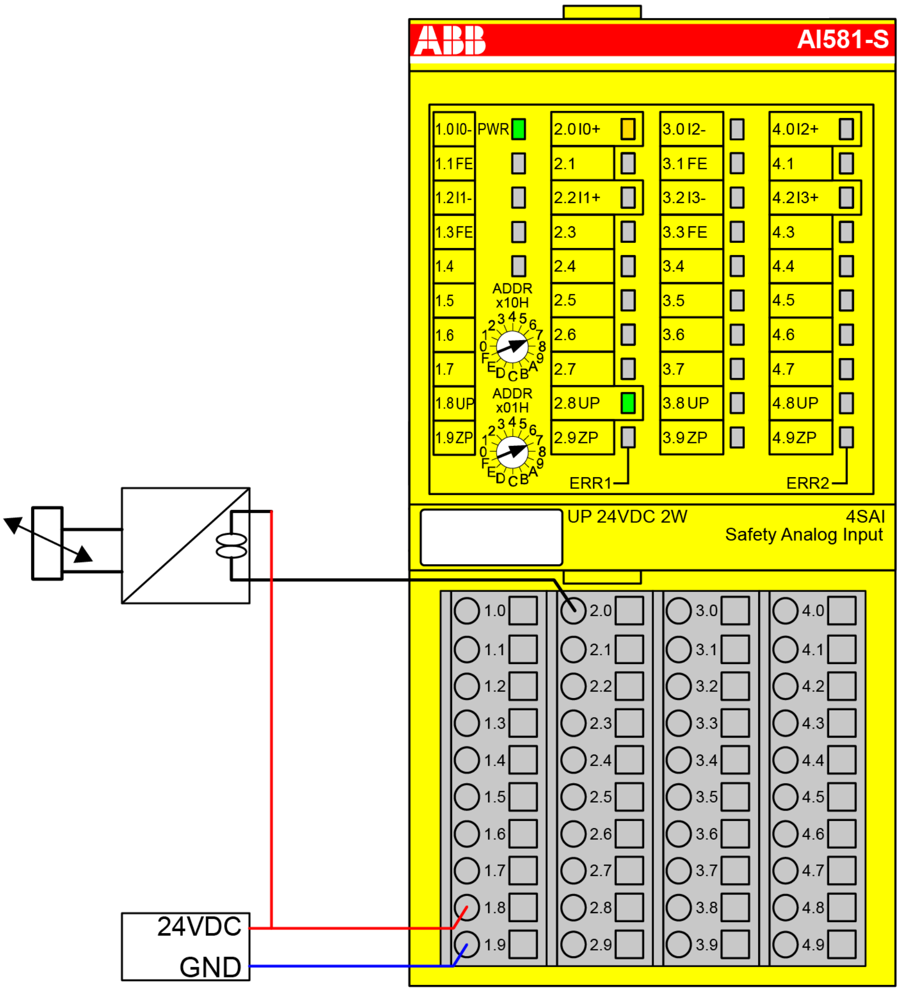


Fig. 61: Circuit example AI581-S, transmitter (4 ... 20 mA)

1) - MTTFd = High, DC = Medium

2) - Max. reachable (ISO 13849-1, IEC 62061) ➔ without error exclusion (you can reach higher levels up to PL e, SILCL 3 with error exclusion)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required) ➔ without error exclusion (you can reach higher levels up to SIL 3 with error exclusion)

2 transmitters (4 ... 20 mA)

2-channel-evaluation	In AI581-S module
SILCL / PL [1), 2)]	SILCL 3 / PL e
SIL[3)]	SIL 3

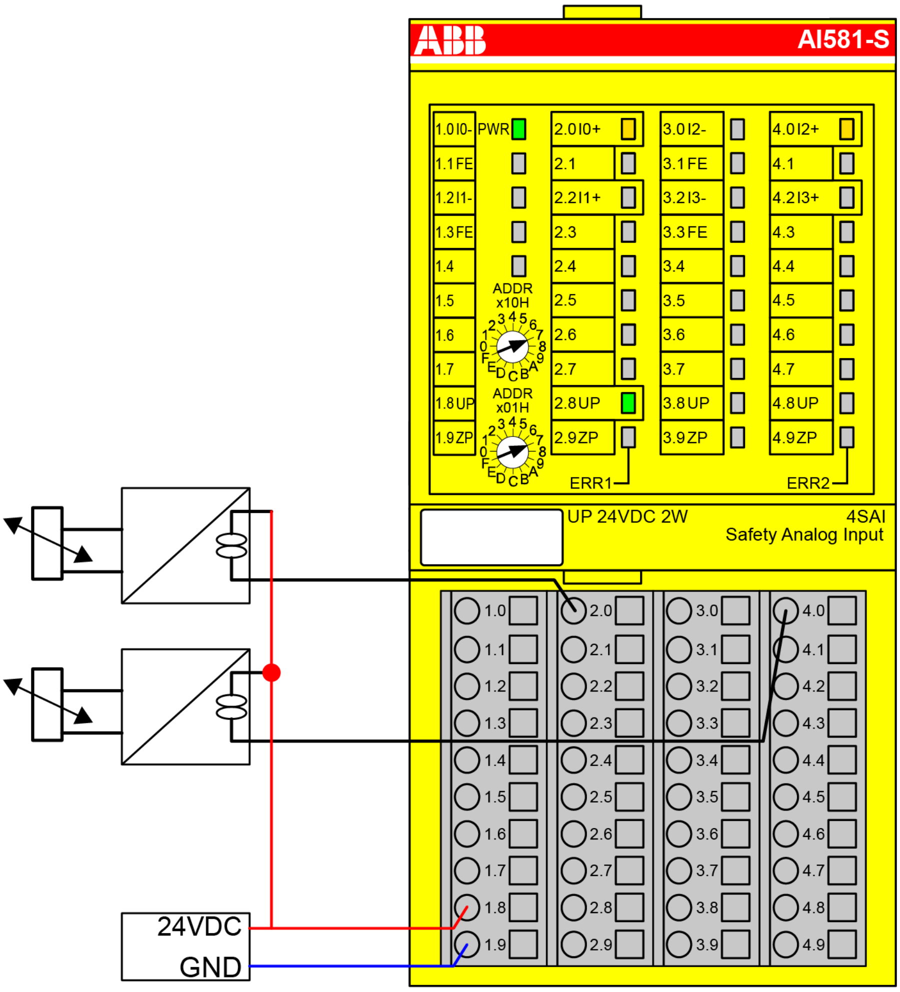


Fig. 62: Circuit example AI581-S, 2 transmitters (4 ... 20 mA)

1) - MTTFd = High, DC = High

2) - Max. reachable (ISO 13849-1, IEC 62061)

3) - Max. reachable SIL acc. IEC 61508 (type A components are required)

3.5.8 LED status display

Table 8: Status display and its meaning

LED	Description	Color	LED = OFF	LED = ON	LED flashes
Inputs 0 ... 3	Analog input	Yellow	Analog input = ca. 0 mA	Input = ON (LED light intensity depends on the input value)	--
	Channel error	Red	No channel error	Channel error	--
UP	Process voltage +24 V DC via terminal	Green	Process supply voltage is missing	Process supply voltage is OK	--
PWR	+3.3 V voltage from I/O bus	Green	+3.3 V I/O bus voltage is not available	+3.3 V I/O bus voltage is available	--

LED	Description	Color	LED = OFF	LED = ON	LED flashes
ERR1	Module error indicator 1	Red	No module error	Module error which leads to a SAFE STOP state	Module passivation and/or acknowledgment request (alternating blinking)
ERR2	Module error indicator 2	Red			

3.5.9 Technical data

> **NOTICE!**
> AI581-S-XC version is available for usage in extreme environmental conditions *Appendix A “System data for AC500-S-XC” on page 367*.

Additional technical data is available in ABB PLC catalog at *www.abb.com/plc*.

Process supply voltage UP

Data	Value	Unit
Connections terminals 1.8 ... 4.8 (UP)	+24	V
Connections terminals 1.9 ... 4.9 (ZP)	0	V
Rated value (-15 %, +20 %, without ripple)	24	V DC
Max. ripple	5	%
Protection against reversed voltage	yes	
Rated protection fuse for UP (fast)	10	A
Electrical isolation	per module	
Mechanisms in which I/Os are processed	periodically refreshed	
Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range, typically	±1	%
Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range, max.	±1.5	%
Maximum signal frequency	70	Hz
Current consumption from UP at normal operation with + 24 V DC (for module electronics)	0.18	A
Inrush current from UP at 30 V (at power up)	0.1	A^2s
Inrush current from UP at 24 V (at power up)	0.06	A^2s

Mounting position

Horizontal or vertical with derating (maximal operating temperature reduced to +40 °C).

Cable length

Data	Value	Unit
Conductor cross section of analog cables	> 0.14	mm^2
Max. analog cable length, shielded	100	m

Cooling

The natural convection cooling must not be hindered by cable ducts or other parts in the switchgear cabinet.

Allowed interruptions of power supply, according to EN 61131-2

Data	Value	Unit
DC supply interruptions	< 10	ms
Time between 2 DC supply interruptions, PS2	> 1	s

Environmental conditions

Data	Value	Unit
Operating temperature*	0 ... +60	°C
Storage temperature	-40 ... +85	°C
Transport temperature	-40 ... +85	°C
Humidity without condensation	max. 95	%
Operating air pressure	> 800	hPa
Storage air pressure	> 660	hPa
Operating altitude	< 2000	m above sea level
Storage altitude	< 3500	m above sea level

* Extended temperature ranges (below 0 °C and above +60 °C) can be supported in special versions of AI581-S ↬ *Appendix A "System data for AC500-S-XC" on page 367*.

Creepage distances and clearances

The creepage distances and clearances meet the overvoltage category II, pollution degree 2.

Power supply units

For the supply of modules, power supply units according to PELV/SELV specifications must be used.

Electromagnetic compatibility

For information on electromagnetic compatibility refer to the latest TÜV SÜD Report ↬ *[2]*.

Mechanical properties

Data	Value	Unit
Degree of protection	IP 20	
Housing	according to UL 94	
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 3.5 mm	2 ... 15	Hz
Vibration resistance acc. to EN 61131-2 (all three axes), continuous 1 g *	15 ...150	Hz
Shock test (all three axes), 11 ms half-sinusoidal	15	g
MTBF	102	years

* Higher values on request

Self-test and diagnostic functions

Start-up and runtime tests: Program flow control, RAM, CPU, ADC, etc.

Dimensions, weight

Data	Value	Unit
W x H x D	67.5 x 76 x 62	mm
Weight (without terminal unit)	~ 130	g

Certifications CE, cUL (further certifications at *www.abb.com/plc*)

3.5.9.1 Technical data of safety analog inputs

DANGER!
Exceeding the permitted process or supply voltage range (< -35 V DC or > +35 V DC) could lead to unrecoverable damage of the system.

Data	Value	Unit
Number of channels per module	4	
Configurability, 1 channel mode	0 ... 20	mA
Configurability, 1 channel mode	4 ... 20	mA
Configurability, 2 channel mode	4 ... 20	mA
Channel input resistance, in active mode	~ 125	Ω
Channel input resistance, in inactive mode	~ 15	kΩ

Distribution of channels into groups 2 groups of 2 channels each.

Data	Value	Unit
Time constant of the input filter	1	ms
Conversion cycle	0.33	ms
Resolution	14	bits
Temperature coefficient ± % of full scale (0 ... 20 mA)	±0.005	%/K
Maximum error at +25 °C ± % of full scale (0 ... 20 mA)	± 0.25	%
Maximum error over full temperature range ± % of full scale (0 ... 20 mA)	± 0.25	%
Value of a LSB (least significant bit)	2.03	μA
Maximum permanent allowed overload (no damage) (self-protected), voltage	32	V DC
Maximum permanent allowed overload (no damage) (self-protected), current	24	mA
Non-linearity (of full scale)	±0.05	%
Sample repetition time	3.3	ms
Input filter characteristics - first order, filter time constant	1	ms
Transition frequency	160	Hz
Overvoltage protection	Yes	

Electrical isolation Against internal supply and other modules.

Input signal indication One LED per channel.

Maximum temporary deviation during specified electrical interference test ± % of full scale

Data	Value	Unit
Deviation during radiated and conducted disturbance	< 0.1	%
Deviation during burst test	max. 0.33	%
Deviation during surge test	up to 50	%
Deviation during electrostatic discharge	no deviation	

Analog input protection

Data	Value
Type of analog input protection	suppressor diode

Cable length

Data	Value	Unit
Max. cable length, shielded	100	m

3.5.10 Ordering data

Type	Description	Part no.
AI581-S	Safety analog input module 4SAI	1SAP 282 000 R0001
AI581-S-XC	Safety analog input module 4SAI, extreme conditions	1SAP 482 000 R0001

3.6 TU582-S safety I/O terminal unit

Elements of the module

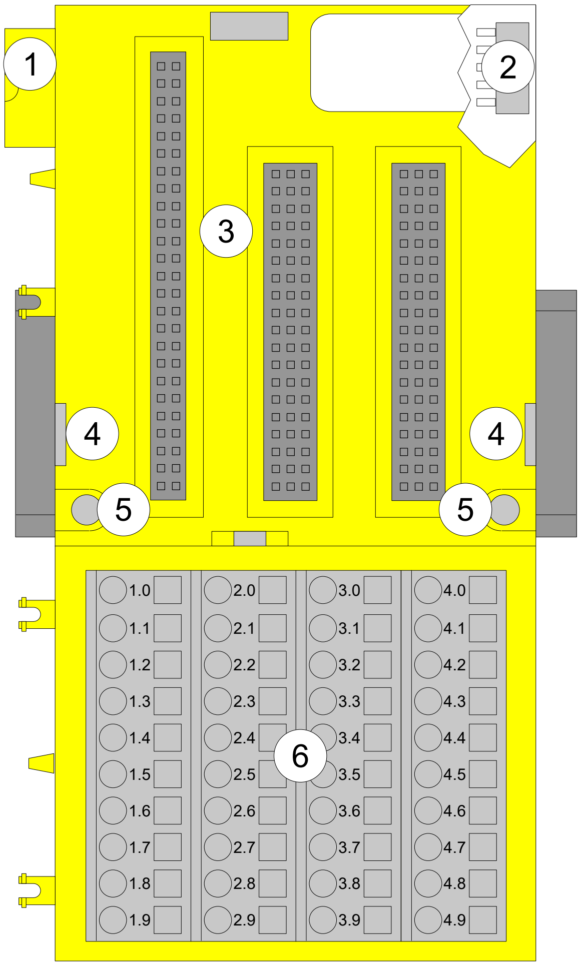


Fig. 63: Safety I/O terminal unit TU582-S (spring-type) for safety I/O expansion modules

1 I/O bus (10-pole, male)
2 I/O bus (10-pole, female)
3 Slot for I/O module
4 With a screwdriver, inserted in this place, adjacent terminal units can be shoved from each other.
5 Holes for wall mounting
6 40 spring terminals (signals and process voltage)

3.6.1 Functionality

The I/O terminal units TU582-S (with spring-type terminals) is specifically designed for use with AC500-S safety I/O modules AI581-S, DI581-S and DX581-S.

The safety I/O modules plug into the I/O terminal unit. When properly seated, they are secured with two mechanical locks. All the electrical connections are made through the terminal unit, which allows removal and replacement of the I/O modules without disturbing the wiring at the terminal unit.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and have always the same assignment, independent of the inserted module:

- Terminals 1.8 to 4.8: Process voltage UP = +24 V DC
- Terminals 1.9 to 4.9: Process voltage ZP = 0 V

The assignment of the other terminals is dependent on the inserted safety I/O module *DI581-S* *DX581-S* *AI581-S*.

3.6.2 Mounting, dimensions and electrical connection

The safety I/O modules can be plugged only on spring-type TU582-S I/O terminal unit. The unique mechanical coding on I/O terminal units prevents a potential mistake of placing the non-safety I/O module on safety I/O terminal unit and the other way around. Basic information on system assembly is shown here. Detailed information can be found in ⇘ *[4]*.

Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204 part 1, by skilled electricians only.

Assembly of TU582-S on DIN rail

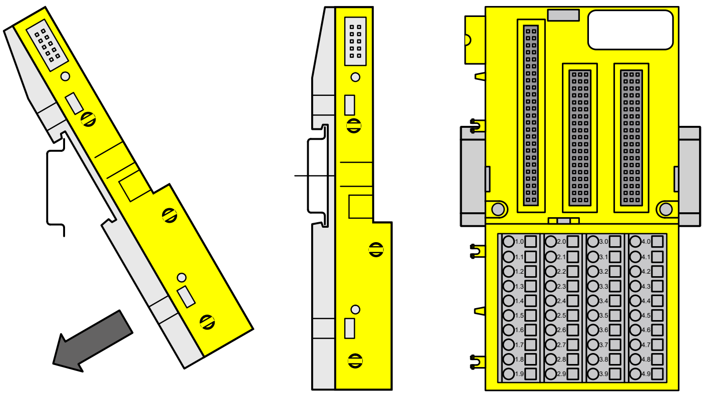
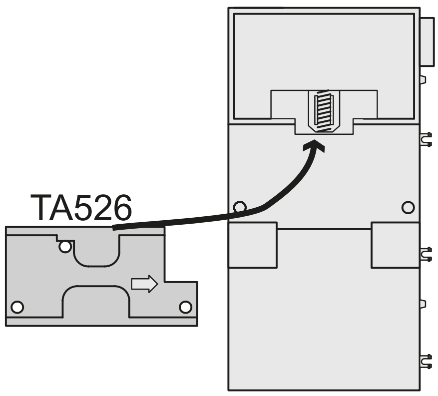


Fig. 64: Assembly instruction for mounting on a DIN rail

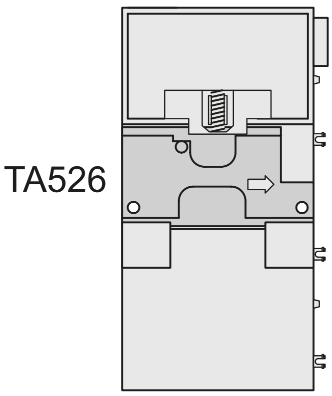
▷ Put the terminal unit on the DIN rail above and then snap-in below.

Assembly of TU582-S with screws

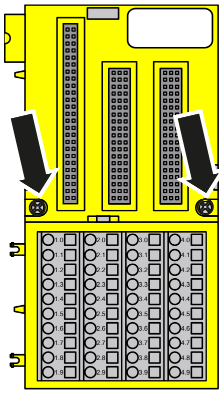


The insertion of the accessories TA526 for wall mounting is essential.

1. Snap TA526 on the rear side of the terminal unit like DIN rails.

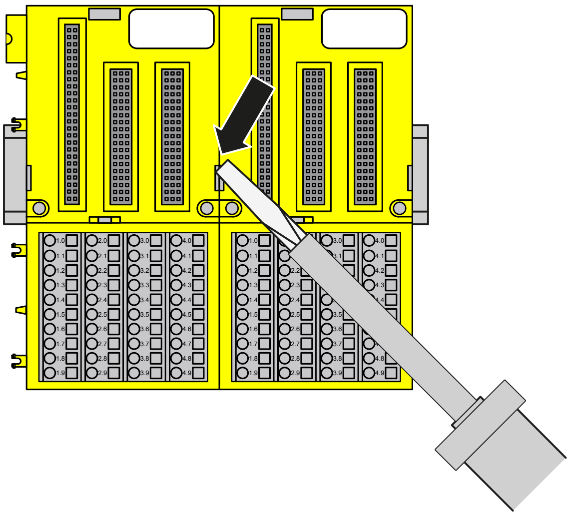


2. Fasten terminal unit with 2 M4 screws (max. 1.2 Nm).

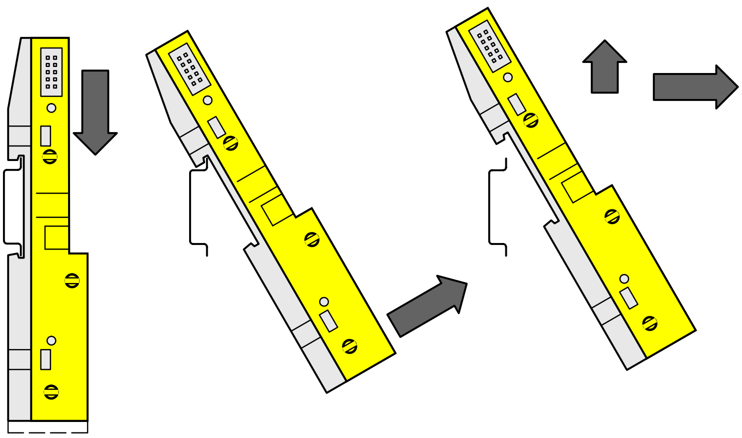


Disassembly of TU582-S

1. Shove the terminal units from each other.



2. Pull down the terminal unit and remove it.



Dimensions

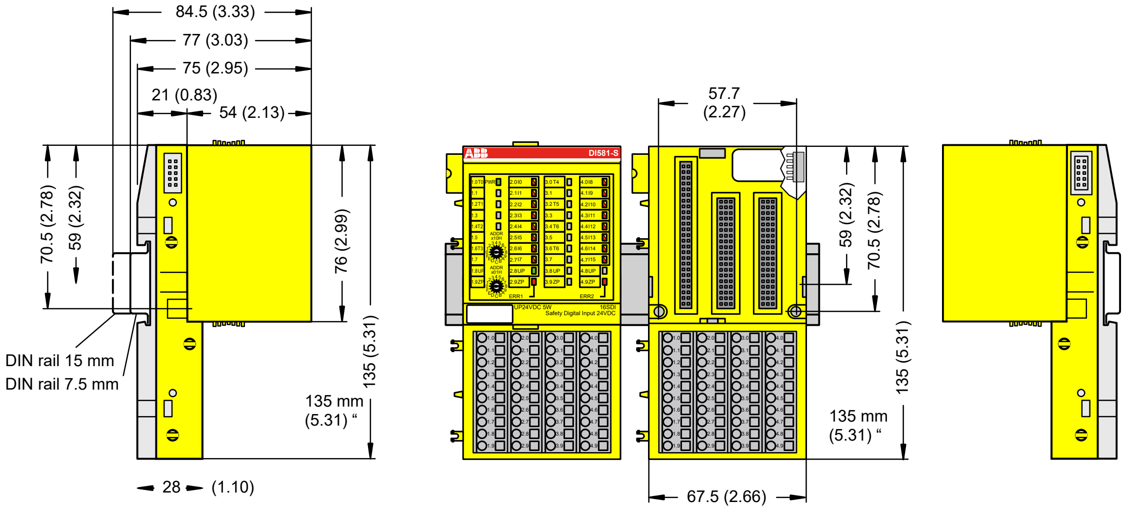


Fig. 65: Dimensions of TU582-S safety I/O terminal unit

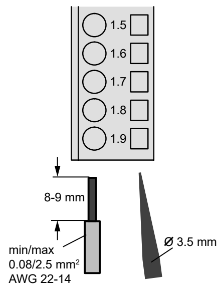


Fig. 66: Spring terminal (screw-driver opens terminal)

3.6.3 Technical data

NOTICE!
TU582-S-XC version is available for usage in extreme environmental conditions
Appendix A "System data for AC500-S-XC" on page 367.

Additional technical data is available in ABB PLC catalog at www.abb.com/plc.

Type

Front terminal, conductor connection vertically with respect to the printed circuit board.

Data	Value	Unit
Number of channels per module	32	
Rated voltage	24	V DC
Max. permitted total current (between the terminals 1.8 ... 4.8 and 1.9 ... 4.9)	10	A

Distribution of channels into groups

4 groups of 8 channels each (1.0 ... 1.7, 2.0 ... 2.7, 3.0 ... 3.7, 4.0 ... 4.7), the allocation of the channels is given by the inserted I/O expansion module.

Mounting position Horizontal or vertical.

Earthing Direct connection to the earthed DIN rail or via the screws with wall mounting.

Conductor

Data	Value	Unit
Conductor cross section, solid	0.08 ... 2.5	mm²
Conductor cross section, flexible	0.08 ... 2.5	mm²
Conductor cross section, with wire-end ferrule	0.25 ... 1.5	mm²
Stripped conductor end, minimum	5	mm
Stripped conductor end	7	mm

Mechanical properties

Data	Value	Unit
Degree of protection	IP 20	
MTBF	2757	years
Weight	~ 200	g

3.6.4 Ordering data

Type	Description	Part no.
TU582-S	Safety I/O terminal unit, 24V DC	1SAP 281 200 R0001
TU582-S-XC	Safety I/O terminal unit, 24V DC, extreme conditions	1SAP 481 200 R0001

4 Configuration and programming

4.1 Overview

4.1.1 Automation Builder

The engineering suite Automation Builder is a platform for configuration and programming of IEC 61131 related applications.

For configuring and programming safety applications, you must use Automation Builder with installed and licensed safety engineering with its safety components (CODESYS Safety programming editor and safety configurator).

The safety concept for safety components in Automation Builder software assures that the programming system works correctly for implementing safety functions in AC500-S, meaning that programming system errors can be detected. The communication between CODESYS Safety and the safety CPU is not a part of the safety loop, but is still subject to checks, for example, a CRC is used during the download of a project in order to verify that the data are transferred correctly and that there is no communication error. The user is responsible to additionally check the version and functionality of his project as well as the proper configuration of safety and non-safety modules.

The Automation Builder safety components allow creating safety applications up to SIL 3 (IEC 61508) / PL e (ISO 13849) safety integrity level.

The compatibility of Automation Builder version is dependent on the used safety and non-safety CPUs *Appendix B.1 “Compatibility with AC500 V2 non-safety CPU” on page 373* *Appendix C.1 “Compatibility with AC500 V3 non-safety CPUs” on page 391*.

4.1.2 Safety engineering

You can easily check your installed and licensed safety engineering version and its safety components. This function is available as of Automation Builder 2.3.0.

☑ Automation Builder is open.

▷ Go to menu *“Help ➔ About ➔ Safety Version Information”*.

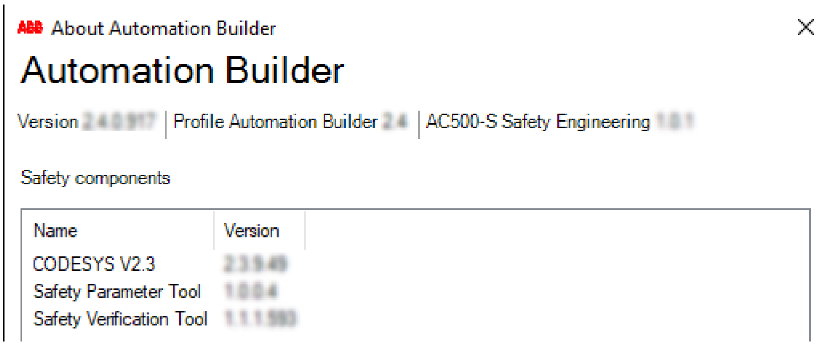


Fig. 67: Information on safety engineering and safety components

If a safety engineering version and the safety components versions are shown, this ensures that you use released and assessed safety components.

The safety components are released independently from Automation Builder releases. After installation of Automation Builder 2.3.0 or higher, the user has to check the safety engineering version *Chapter 4.2 “Workflow” on page 134*.

NOTICE!
If no safety engineering and no safety components are shown, redo the Automation Builder installation once again and make sure you have activated the appropriate license. If the error persists, contact ABB technical support.

4.1.3 Safety measures

A complete check of program logic and configuration must be performed to verify that logic correctly and fully addresses the functional and safety requirements in your safety application specification. Each time you make a modification, re-check project data.

DANGER!
For the initial start-up of a safety CPU or after a modification of the application program or configuration, the safety of the entire system must be checked by a complete functional test, which includes also the check of the correct coding of the safety application based on the functional specification.

4.1.4 Protection against unintended modifications

Protection mechanisms are integrated in the safety CPU and in Automation Builder with safety features to prevent unintentional or unauthorized modifications to the safety system:

- A modification of the safety application program generates a new boot project CRC version number.
- The user must be logged in to the safety CPU to access its operating options.
- Requirements of safety and other relevant application standards regarding protection against manipulations must be observed. The authorization of employees and the necessary protection measures are the responsibility of the operator in charge.

Any unauthorized access to safety CPU and safety program can be protected by several passwords *Chapter 4.3.3 “Creation of new project and user management” on page 135*.

4.2 Workflow

The engineering workflow presented in this chapter describes only the steps needed to instantiate, configure and program safety modules and those non-safety modules which are a part of the "black channel" *[3]* in the safe communication part. All other non-safety modules are separately covered in *[4]*. For more details on these steps refer to *Chapter 4.3 “System configuration and programming” on page 135*.

AC500-S system configuration and programming workflow

1. Install Automation Builder, as described in the installation guide.
2. Activate a license.
3. As of Automation Builder 2.3.0: Check that the safety engineering and the safety components are available *Chapter 4.1.2 “Safety engineering” on page 133*.
4. Create a new project and configure user management to limit access to safety modules and their configuration to safety personnel only.
5. Install GSDML files to be able to configure 3rd party PROFIsafe F-Devices (optional step).

6. Instantiate safety modules and non-safety modules, which are a part of the "black channel" for safe communication, and do a proper configuration of those. Define variable names for input, output and PROFIsafe signals and pay attention to CODESYS Safety programming guidelines to define proper variable names 🖙 *Chapter 4.4 “CODESYS Safety programming guidelines” on page 179*.
7. Write your safety application program and pay attention to system start-up procedure.
8. Check your program and system configuration. Use the SCA tool for static code analysis of your program 🖙 *Chapter 4.5 “Safety code analysis tool” on page 188*. Follow the procedures for checking your configuration 🖙 *Chapter 4.3.7 “Checking of program and system configuration” on page 166*.

4.3 System configuration and programming

In this chapter, we provide a step-by-step explanation on how to configure and program AC500-S safety PLC.

4.3.1 Installation

▷ Install Automation Builder, as described in its installation guide.

4.3.2 License activation

Automation Builder 2.0.2 (or higher)

1. Order DM220-FSE or DM221-FSE-NW add-on with part numbers 1SAS010020R0102 and 1SAS010021R0102.
2. Activate license on your PC following license activation instructions.

Automation Builder up to 1.2.4

1. Order PS501-S license with part number 1SAP198000R0001.
2. Activate license on your PC following license activation instructions.

4.3.3 Creation of new project and user management

Create a new project and configure user management to permit access to safety modules and their configuration to safety personnel only.

1. Use *“New project...”* menu item in Automation Builder to create a new project.
2. Select a non-safety AC500 CPU in the menu. Make sure that you select the right ones supporting safety CPUs 🖙 *Appendix B.1 “Compatibility with AC500 V2 non-safety CPU” on page 373* 🖙 *Appendix C.1 “Compatibility with AC500 V3 non-safety CPUs” on page 391*.

> **!** **NOTICE!**
> Pay attention to non-safety CPU settings 🖙 *Appendix B.3 “AC500 V2 non-safety CPU parameters configuration” on page 382* 🖙 *Appendix C.3 “AC500 V3 non-safety CPU parameters configuration” on page 397*.

3. To create new users and maintain existing ones, go to *“Project ➔ Project Settings...”*.

> **NOTICE!**
> In all newly created Automation Builder projects, there is a default user "Owner" with an empty password. This is a project administrator. The project administrator is responsible to create a new password for user "Owner" and, in addition, create dedicated safety and non-safety users based on your project organization demands.

Only members of safety group are allowed to modify safety modules, change their configuration, etc. By default, no users without proper log-in and access rights can access safety modules.

Access to safety CPU and safety program can be protected by three passwords.

- Password for the safety CPU
- Password for the safety program in CODESYS Safety
  - max. 200 characters
  - allowed characters: (A-Z) (a-z) (0-9) Ä Ö Ü ä ö ü ß # § % ° ^ + - & _ ! @ ´ ~ * | ( ) { } [ ] , ; . : <> = / ' ?
- Password for safety modules and their configuration data in Automation Builder with safety features

Project administrator is allowed to use all available user management features to find the best suitable user setup with appropriate rights *[4]*.

> **DANGER!**
> It is the responsibility of project administrator to setup a proper user management for the given safety application project to avoid unauthorized access to safety modules.
>
> Passwords for users with safety group membership shall be properly selected (at least 8 symbols are recommended with a combination of numbers and letters). An access to passwords must be strictly controlled.
>
> Make sure that you set *“Deny”* permission for proper users and groups (e.g., Everyone) through menu *“Project ➔ User Management ➔ Permissions...”* to avoid unauthorized creation of new users in the safety group.

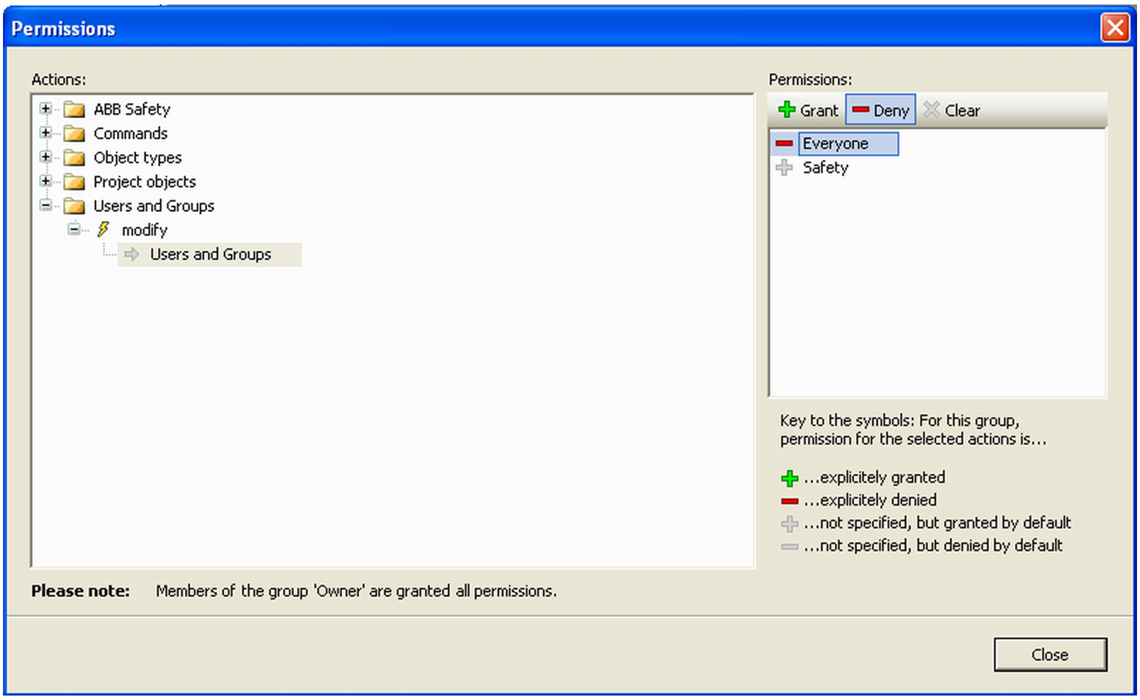


Fig. 68: Permissions for user and user groups

4.3.4 Working with PROFINET/PROFIsafe F-Devices

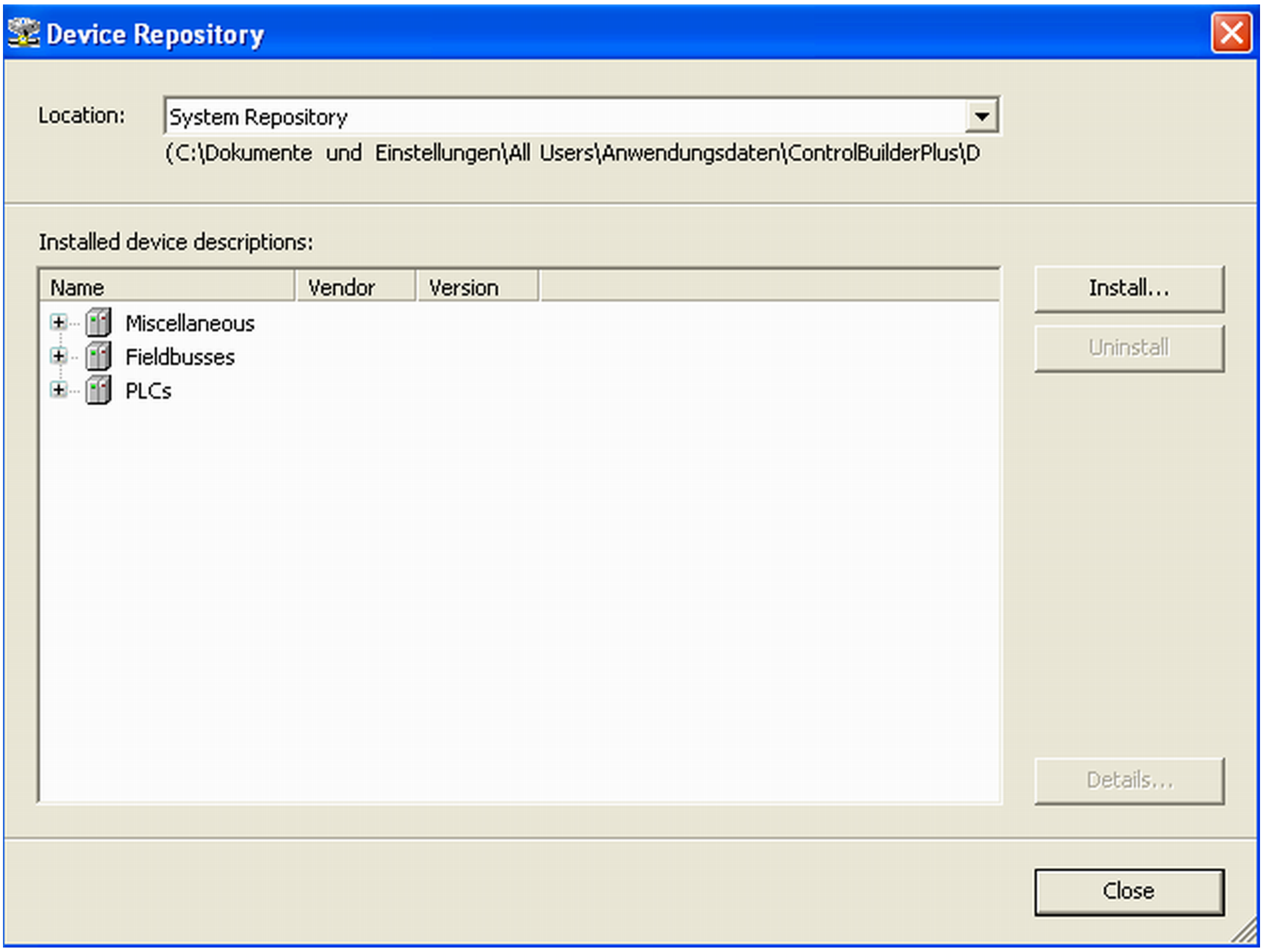
You have to install GSDML files to be able to configure 3rd party PROFIsafe F-Devices.

In order to use 3rd party F-Devices with AC500-S safety PLC, the safety devices must be on the PROFINET and support the PROFIsafe bus profile in V2 mode *[3]*. The basis for configuring all (safety and non-safety) PROFINET devices is the specification of the device in the GSDML file (generic station description markup language).

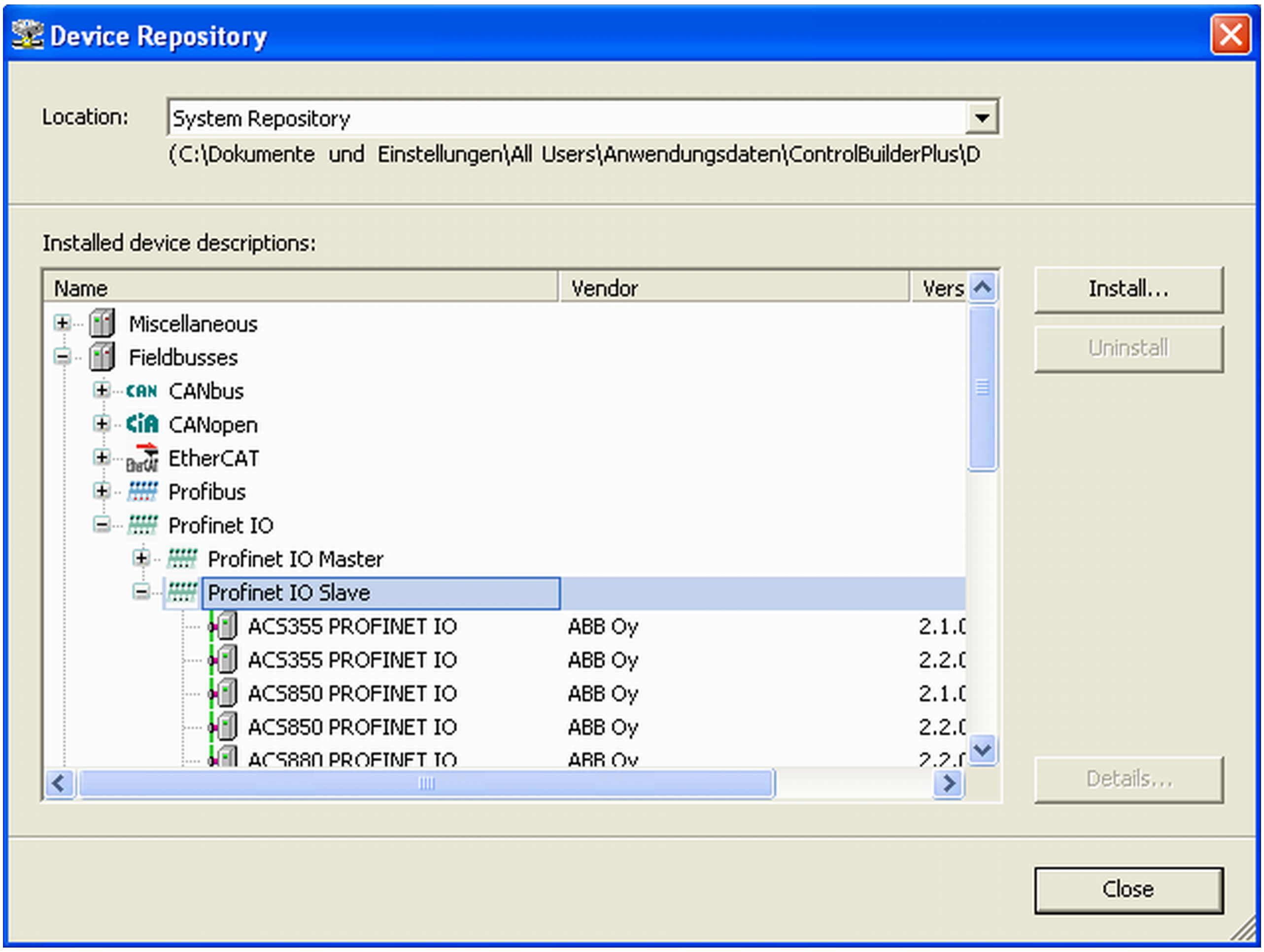
I/O device properties are saved in the GSDML file. For PROFINET/PROFIsafe devices, portions of the GSDML file data are protected by a CRC *[3]*. GSDML files are supplied by the device manufacturers.

> **NOTICE!**
> Please contact ABB technical support for details on supported GSDML file versions. It depends on the version of your installed Automation Builder.

1. To install GSDML file, go to *“Tools ➔ Device Repository...”* menu.



2. Press *[Install...]* button to pick-up a GSDML file and install it.
   ⇨ After successful installation, new devices are shown in *"Device Repository"* under *"Profinet IO"* object.



4.3.5 Instantiation and configuration of safety modules / definition of variable names

Instantiate safety and non-safety modules, which are a part of the "black channel" for safe communication and do a proper configuration of those. Define variable names for input, output and PROFIsafe signals and pay attention to CODESYS Safety programming guidelines to define proper variable names.

1. Select one of four slots available for communication modules and safety CPU and instantiate a safety CPU on it. Note, that the slot number shall be the same as the physical slot number on which safety CPU is attached.
2. Double-click on the safety CPU and set its parameters, as needed.

> **NOTICE!**
> Pay attention to the safety CPU parameter *"Enable debug"*. If this parameter is set to *"OFF"*, then no new boot project can be loaded to the safety CPU.
>
> If a new boot project has to be loaded to the safety CPU, then, in advance, a new boot project with *"Enable debug"* parameter set to *"ON"* for the safety CPU shall be loaded to non-safety CPU. After the reboot of non-safety CPU, a new boot project can be loaded to the safety CPU.
>
> Note that the following PLC browser commands are supported on the safety CPU only if *"Enable debug"* parameter is set to *"ON"* ➡ *list of all PLC browser commands*:
>
> – resetprg - reset safety CPU program
> – resetprgorg - reset safety CPU program original
> – setpwd - set safety CPU login password
> – delpwd - delete safety CPU login password
> – delappl - delete user program
> – deluserdat - delete user data segments

Depending on the used non-safety AC500 CPU the parameter for cycle time is called in a different way. The meaning of both parameters is identical.

With AC500 V2 non-safety CPU: *"Min update time"*

With AC500 V3 non-safety CPU: *"Update cycle time"*

Note, that the parameter *"Min update time"* / *"Update cycle time"* influences the safety function response time. The smaller the value is, the faster the safety function response time will be ➡ *Chapter 5.1 "Overview" on page 325*. However, at the same time, the load on non-safety CPU increases with smaller values of *"Min update time"* / *"Update cycle time"*.

> **DANGER!**
> Big values (e.g., > 10 ms) of *"Min update time"* / *"Update cycle time"* parameter increase the chance of not delivering input pulse signals with a length < *"Min update time"* / *"Update cycle time"* value to the safety CPU.

3. To have remote stations in the system, we can instantiate PROFINET IO controller communication module CM579-PNIO, for example, in slot 2. Note that PROFINET is the only bus which is supported for PROFIsafe communication in AC500-S safety PLC.
4. Now, select newly created CM579-PNIO module and instantiate the required number of PROFINET modules, e.g., CI501-PNIO, CI502-PNIO, etc. or any 3rd party PROFINET modules previously imported in the *"Device Repository"* using GSDML files.

   Details on how to set proper PROFINET device names and IP addresses can be found in ➡ *[4]*.
5. On *"IO_Bus"* object, one can instantiate up to 10 I/O modules (safety or non-safety ones) located centrally on the non-safety CPU.
6. Similarly, up to 10 I/O modules (safety and non-safety) can be instantiated on any ABB PROFINET IO device.

GSDML file defines the maximum number of supported modules on 3rd party PROFINET IO devices.

Parameters of safety I/O modules can be set using double-click on those modules. Each module has two types of parameters: F-Parameters and iParameters.

F-Parameters are parameters which were specially defined by PROFIsafe group *[3]* to realize safe device communication and parameterisation. F-Parameter names are the same for all F-Devices (ABB and 3rd party devices). The most important of them for end-users are F_SIL, F_Dest_Add, F_Source_Add, F_WD_Time and F_iPar_CRC. F_WD_Time parameter is further used in safety function response time calculations *Chapter 5.3 "Safety function response time" on page 325*. F_Dest_Add shall be the same address as the one set on the physical safety I/O device.

> **NOTICE!**
> Make sure that F_Dest_Add is set unique for all F-Devices, otherwise no valid safety configuration can be generated.
>
> Decimal or hexadecimal number with a prefix 16# or 0x can be used to set F_Dest_Add in Automation Builder.

F_iPar_CRC is a special parameter which is used for a safe transfer of iParameters to F-Devices. F_iPar_CRC is calculated outside F-Parameter editor and, thus, has to be manually copied from *"Checksum iParameter"* field and pasted to F_iPar_CRC field after pressing *[Calculate]* button for the given F-Device.

Note, that F_iPar_CRC has to be recalculated for AC500-S safety I/O modules also if F_Dest_Add is changed, because F_Dest_Add is also invisibly transported as iParameter to AC500-S safety I/O modules. It is needed in AC500-S safety PLC for further comparison of the physical PROFIsafe address value on the safety I/O device and one configured in the engineering environment.

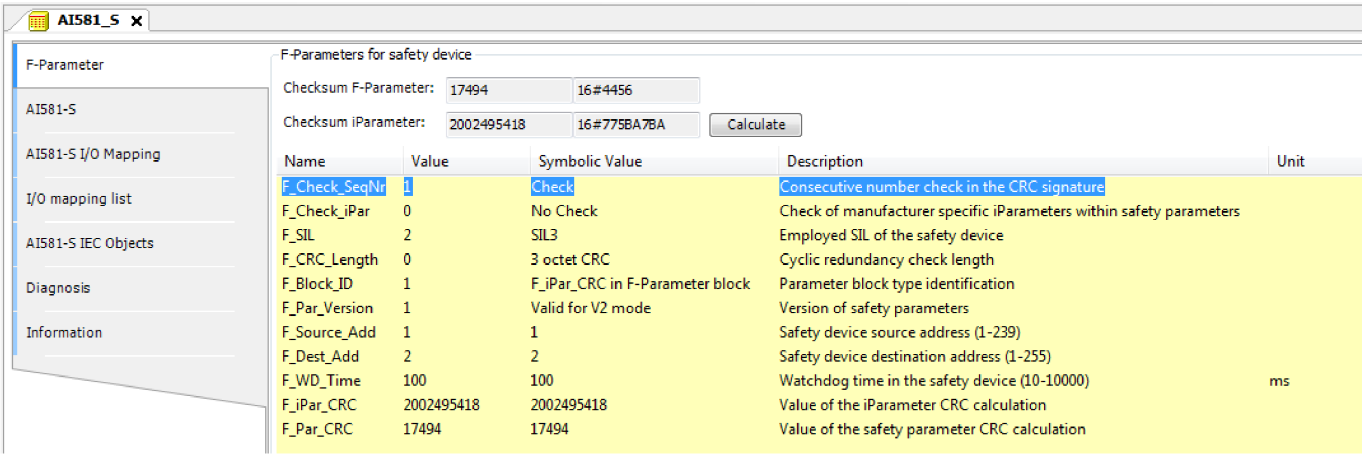


Fig. 69: F-Parameters settings

Table 9: F-Parameters of AC500-S safety modules

F_Parameter	Definition	Allowed values	Default value
F_Check_SeqNr	This parameter defines whether the consecutive number shall be included in the CRC2. PROFIsafe V2-mode ⇘ *[3]*: consecutive number has to be always included in CRC2 generation. Note: F_Check_SeqNr is not shown in the F-Parameter configuration for SM560-S-FD-1 and SM560-S-FD-4.	"No Check" = 0 "Check" = 1	"Check" = 1
F_Check_iPar	Manufacturer-specific use within homogeneous systems	"No Check" = 0 "Check" = 1	"No Check" = 0
F_SIL	Different safety functions using safety-relevant communication may require different safety integrity levels. The F-Devices are able to compare their own assigned SIL with the configured SIL (F_SIL). If it is higher than the SIL of the connected F-Device, the "device failure" status bit is set and a safe state reaction is triggered ⇘ *[3]*.	"SIL1" = 0 "SIL2" = 1 "SIL3" = 2 "NoSIL" = 3	"SIL3" = 2
F_CRC_Length	Depending on the length of the F I/O data (12 or 123 octets) and the SIL level, a CRC of 2, 3, or 4 octets is required	"3 octet CRC" = 0 "2 octet CRC" = 1 Not supported by SM560-S: "4 octet CRC" = 2	"3 octet CRC" = 0 (AC500-S safety I/O modules can work only with "3 octet CRC")
F_Block_ID	Type identification of parameters	"No F_iPar_CRC within F-Parameter block" = 0 "F_iPar_CRC within F-Parameter block" = 1	"F_iPar_CRC within F-Parameter block" = 1 for Safety I/Os (AC500-S safety I/O modules can work only with this default value) "F_iPar_CRC within F- Parameter block" = 0 for SM560-S-FD-1 and SM560-S-FD-4
F_Par_Version	Version number of the F-Parameter set	"Valid for V1-mode" = 0 "Valid for V2-mode" = 1	"Valid for V2-mode" = 1 (AC500-S safety I/O modules can work only with this default value)

F_Parameter	Definition	Allowed values	Default value
F_Source_Add	F-Host source address. The F_Source_Add parameter is a logical address designation that can be assigned freely but unambiguously. **F_Source_Add shall not be equal to F_Dest_Add for the given F-Device.**	[1 - 511] for SM560-S-FD-1 and SM560-S-FD-4 [1 - 239] for AC500-S safety I/O modules [1 - 65534] for 3rd party PROFIsafe F-Devices (if no limitations of F_Source_Add are defined by the manufacturer) 0 and 65535 is not allowed.	1
F_Dest_Add	The unique F-Device address which will be compared with the set hardware switch address in F-Device. The F_Dest_Add parameter is a logic address designation that can be assigned freely but unambiguously.	[1 - 255] for AC500-S safety I/O modules. For SM560-S-FD-1 and SM560-S-FD-4: ● F_Dest_Add = Address Switch Value (1 - 239) * 100 + F-Device instance no. (0..31). ● Addresses switch values [240 - 255] are reserved for system functions.	2 for safety I/O modules 100 for SM560-S-FD-1 or SM560-S-FD-4
F_WD_Time	Watchdog time in ms for receipt of the new valid telegram	[10 - 10000]	ABB F-Devices: 100 3rd party F-Devices: according to GSDML file
F_iPar_CRC	CRC over iParameters (manufacturer-specific) of F-Devices (safety I/Os).	[0 - 4294967295] Hex [0 - FFFFFFFF]	Dependent on the module type
F_Par_CRC	CRC1 signature calculation across the F-Parameters	[0 - 65535] Hex [0 - FFFF]	Dependent on the module type

iParameters are individual F-Device parameters which are transferred to F-Devices with a proper F_iPar_CRC parameter.

NOTICE!

AC500-S PROFIsafe F-Host implementation does not support or only partially supports the following PROFIsafe conformance class ⇨ *[3]* functions:

– Communication function block set RDREC, WRREC, RDIAG and RALRM, as defined in ⇨ *[12]*.
– iPar server services.
– Data types: Unsigned32, Integer32 and real data types like Float32, etc.
– Tool calling interface, as defined in ⇨ *[3]*.

NOTICE!

After changing iParameters, you have to go to *"F-Parameter"* tab, re-calculate iParameter CRC and paste it to F_iPar_CRC F-Parameter row. Otherwise, the new parameter set will not be accepted by the F-Device because F_iPar_CRC will not be a valid one for a given iParameter set.

As for 3rd party F-Devices coming from GSDML files, **one has no *"Checksum iParameter"* feature,** because Automation Builder does not know a specific algorithm used for F_iPar_CRC calculation in 3rd party devices. One has to calculate F_iPar_CRC using a special tool delivered by the F-Device manufacturer for engineering its F-Devices.

Another option is to contact the vendor of the F-Device and ask for F_iPar_CRC value for the given F-Device iParameter. As soon as F_iPar_CRC is available for the given 3rd party F-Device, one can paste it to the F_iPar_CRC row in F-Parameter editor.

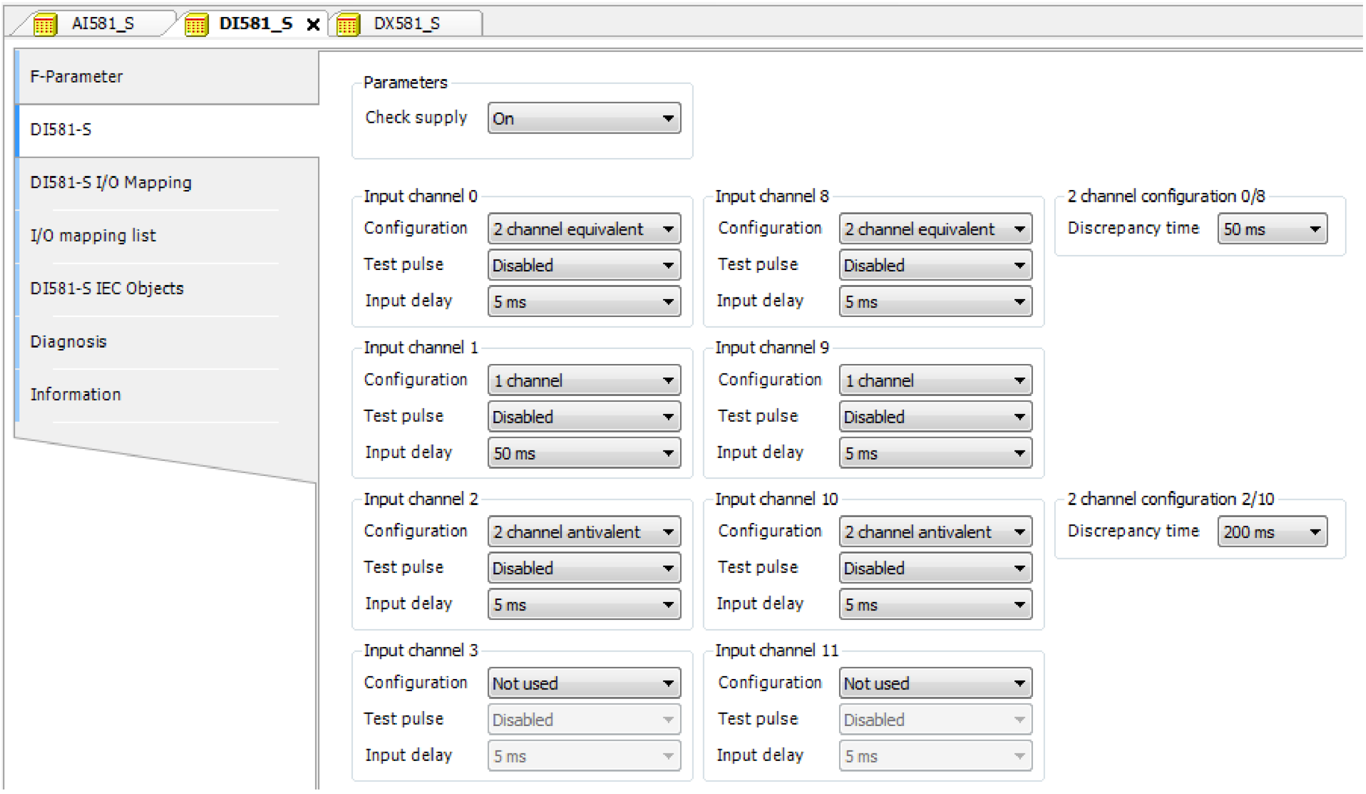


Fig. 70: Examples of iParameter settings for DI581-S safety module; all input channels are paired as "Channel X with Channel X + 8"

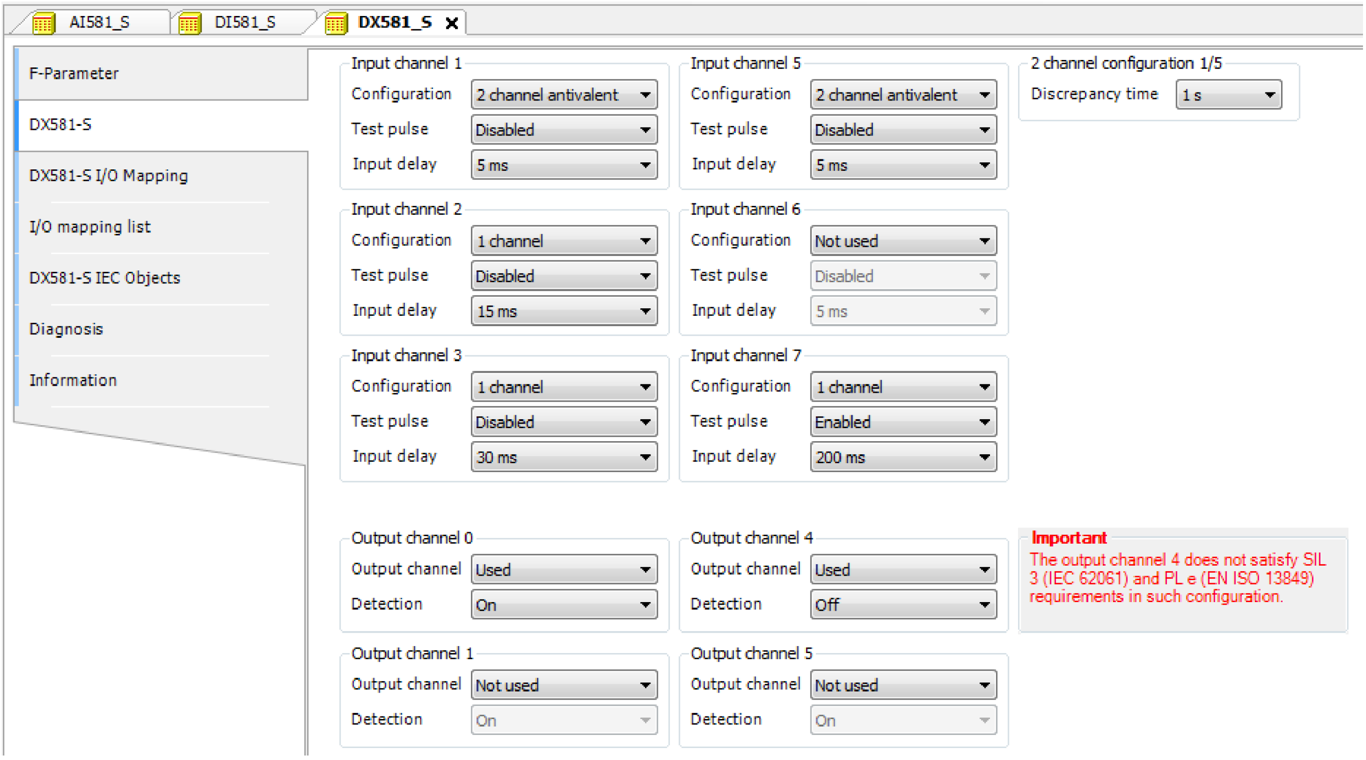


Fig. 71: Examples of iParameter settings for DX581-S safety module; input channels are paired as "Channel X with Channel X + 4"

DANGER!

If for one of the output channels you set Detection = OFF, the warning appears that the output channel does not satisfy SILCL 3 (IEC 62061) and PL e (ISO 13849-1) requirements in such condition. Two safety output channels may have to be used to satisfy required SIL or PL level.

The parameter "Detection" was created for customers who want to use safety outputs of DX581-S for SILCL 1 (or maximum SILCL 2 under special conditions) or PL c (or maximum PL d under special conditions) safety functions and have less internal DX581-S pulses visible on the safety output line. Such internal pulses could be detected as LOW signal by, for example, drive inputs, which would lead to unintended machine stop.

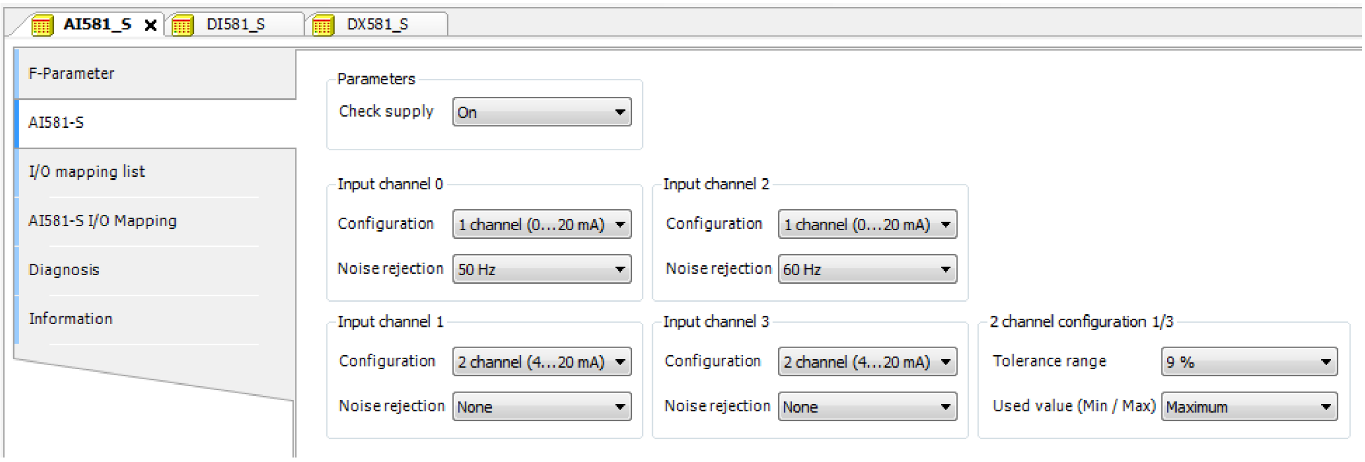


Fig. 72: Examples of iParameter settings for AI581-S safety module; input channels are paired as "Channel X with Channel X + 2"

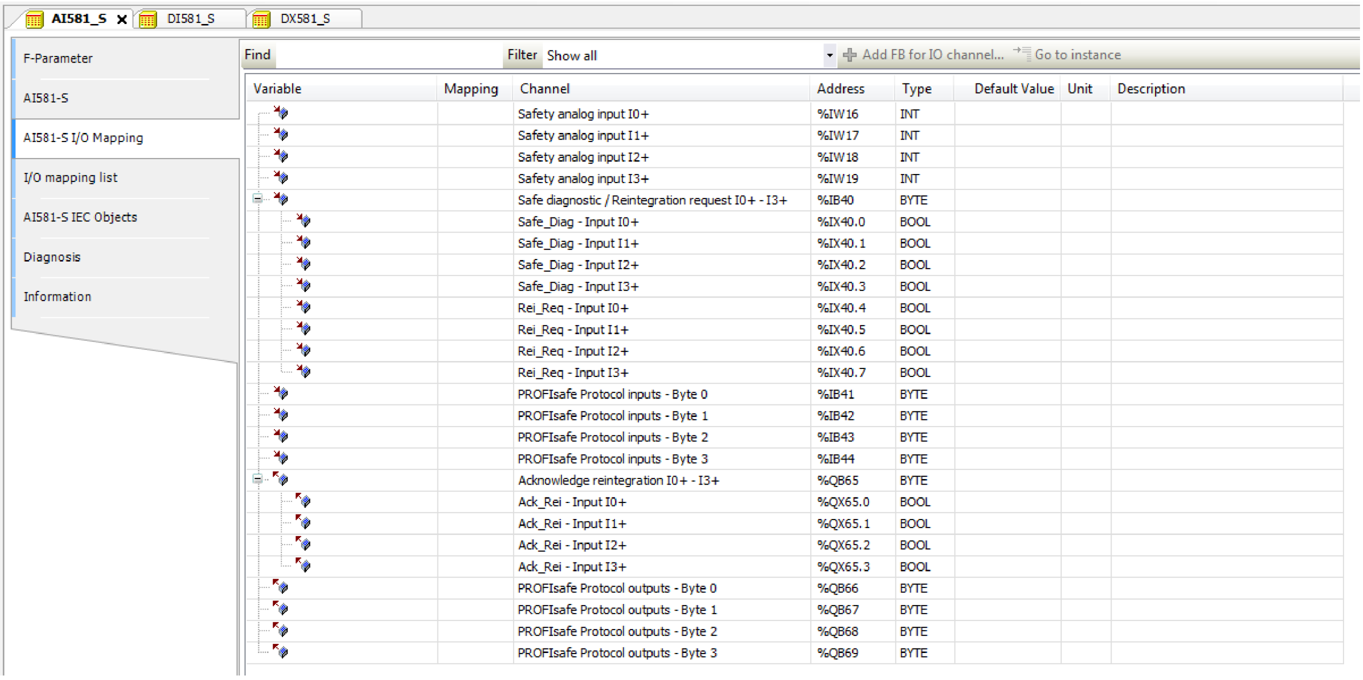
DANGER!

One can also use generic device configuration view from *“DI581-S Parameters”*, *“DX581-S Parameters”* or *“AI581-S Parameters”* tab to edit module and channel parameters. **However, change of safety I/O parameters using generic device configuration view is not recommended** due to potential user mistakes during the parameter setting using integer numbers.

Furthermore, each F-Device has a special *“I/O Mapping”* tab in which variable names for input and output signals, PROFIsafe diagnostic bits, etc. can be defined.

DANGER!

If data types like Unsigned16, Unsigned32, Integer16, Integer32 or Float32, which require more than one byte, are used in PROFIsafe data, note the following. The byte order in such data types depends on the used PROFIsafe device endianness and selected AC500 non-safety CPU type. AC500 V2 non-safety CPU supports big-endian. AC500 V3 non-safety CPU supports little-endian. Make sure that the symbolic variables are mapped properly and the delivered safety data is correctly represented in your safety application.



Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
		Safety analog input I0+	%IW16	INT			
		Safety analog input I1+	%IW17	INT			
		Safety analog input I2+	%IW18	INT			
		Safety analog input I3+	%IW19	INT			
		Safe diagnostic / Reintegration request I0+ - I3+	%IB40	BYTE			
		Safe_Diag - Input I0+	%IX40.0	BOOL			
		Safe_Diag - Input I1+	%IX40.1	BOOL			
		Safe_Diag - Input I2+	%IX40.2	BOOL			
		Safe_Diag - Input I3+	%IX40.3	BOOL			
		Rei_Req - Input I0+	%IX40.4	BOOL			
		Rei_Req - Input I1+	%IX40.5	BOOL			
		Rei_Req - Input I2+	%IX40.6	BOOL			
		Rei_Req - Input I3+	%IX40.7	BOOL			
		PROFIsafe Protocol inputs - Byte 0	%IB41	BYTE			
		PROFIsafe Protocol inputs - Byte 1	%IB42	BYTE			
		PROFIsafe Protocol inputs - Byte 2	%IB43	BYTE			
		PROFIsafe Protocol inputs - Byte 3	%IB44	BYTE			
		Acknowledge reintegration I0+ - I3+	%QB65	BYTE			
		Ack_Rei - Input I0+	%QX65.0	BOOL			
		Ack_Rei - Input I1+	%QX65.1	BOOL			
		Ack_Rei - Input I2+	%QX65.2	BOOL			
		Ack_Rei - Input I3+	%QX65.3	BOOL			
		PROFIsafe Protocol outputs - Byte 0	%QB66	BYTE			
		PROFIsafe Protocol outputs - Byte 1	%QB67	BYTE			
		PROFIsafe Protocol outputs - Byte 2	%QB68	BYTE			
		PROFIsafe Protocol outputs - Byte 3	%QB69	BYTE			

Fig. 73: Example with AI581-S module for variable mapping

It is also valid for DX581-S and DI581-S safety modules; the only difference is the number of input and output channels. Each process channel (Input 0 - Input 3 for AI581-S) has additionally the following bits:

- one bit for safe diagnostic (Safe_Diag bit) to be able to differentiate if the process value is the real process state or "0" value due to channel or module passivation.
- one bit Rei_Req for channel reintegration request, which can be used in the safety application program as a signal that external error (e.g., sensor wiring error) was fixed and the channel can be reintegrated in the safety control. Higher overall system availability can be expected for end-customers, because they can selectively decide which channels have to be acknowledged and which not.
- one bit Ack_Rei for channel reintegration if the error was fixed (e.g., external sensor wiring was corrected). One can also define one variable as a BYTE for all Ack_Rei bits and use 0xFF value to acknowledge all errors at once.

NOTICE!
When you define variable names for input signal, output signal and other safety signals, pay attention to CODESYS Safety programming guidelines *Chapter 4.4 “CODESYS Safety programming guidelines” on page 179*.

NOTICE!
Only BYTE data type is supported instead of WORD for safety data of DI581-S module when AC500 V3 non-safety CPU is used. It is needed to overcome the endianness, which is different between AC500 V2 non-safety CPU (big-endian) and AC500 V3 non-safety CPU (little-endian). This shall be considered when safety project is migrated from AC500 V2 to V3 non-safety CPU.

4.3.6 Programming of AC500-S safety CPU

Write your safety application program and pay attention to system start-up procedure.

NOTICE!

How to create, configure, modify and download a valid CODESYS boot project for non-safety CPUs is described in *[4]*.

To avoid unexpected configuration errors, as a first step, download a valid CODESYS PLC project to non-safety CPU. As a second step, download a CODESYS Safety PLC project to the safety CPU.

1. Program and download a valid project to non-safety CPU.
2. Start CODESYS Safety by double-clicking safety application node, e.g., *"AC500_S"*.

   ⇨ Before CODESYS Safety is started, you may be asked to update your CODESYS configuration. It is needed to transfer the updated configuration data (e.g., variable names, etc.) to CODESYS Safety.

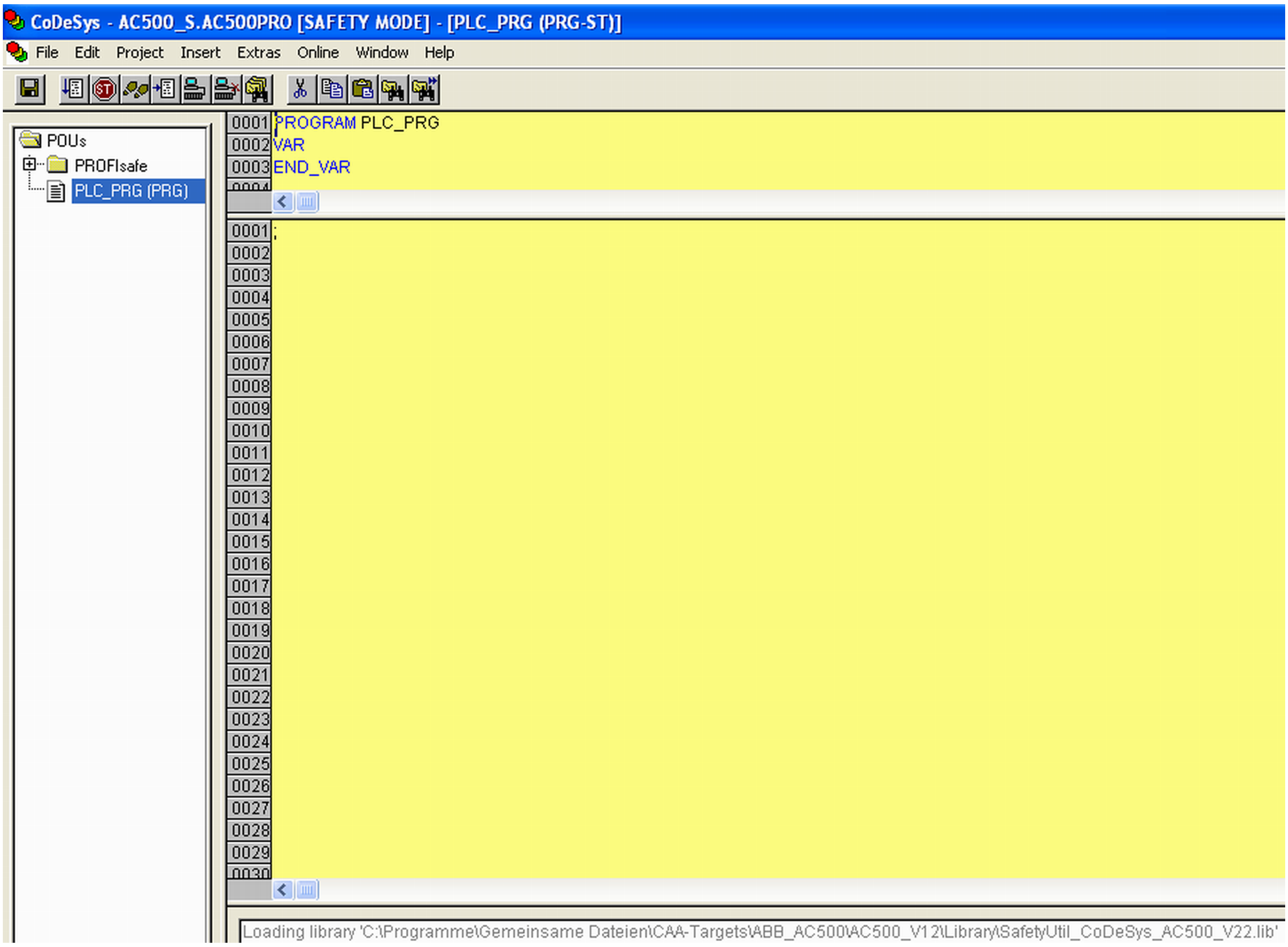


Fig. 74: CODESYS Safety

DANGER!

Make sure that when CODESYS Safety is started, the following properties can be observed:

– Yellow background
– SAFETY MODE is visible in the title bar

NOTICE!

When CODESYS Safety is started for the first time in the Automation Builder project, you will be asked to manually confirm included safety library identification data (version number and CRC). After this, safety library identification data are saved in the project.

If you change the safety library content and replace it on your hard disk, the next time you start CODESYS Safety you will be informed that one of the safety libraries changed. **In the properties window for safety libraries you will still observe an initially saved CRC value.** However, when you compile the project, you will get a CRC error message because of the changed library and the project will not be compiled.

To compile the project successfully, manually delete the selected safety library and add a new safety library with a new CRC. The new safety library with new CRC will be accepted and no compilation error will be shown.

3. Define your user management for CODESYS Safety.

All user management features of CODESYS Safety are available for project administrator ⇘ *[4]*.

The project administrator has to set a user password for newly created CODESYS Safety project. Go to *"Project ➔ User Group Passwords..."* and set the password for Level 0 User Group, which shall represent users from safety user group in Automation Builder.

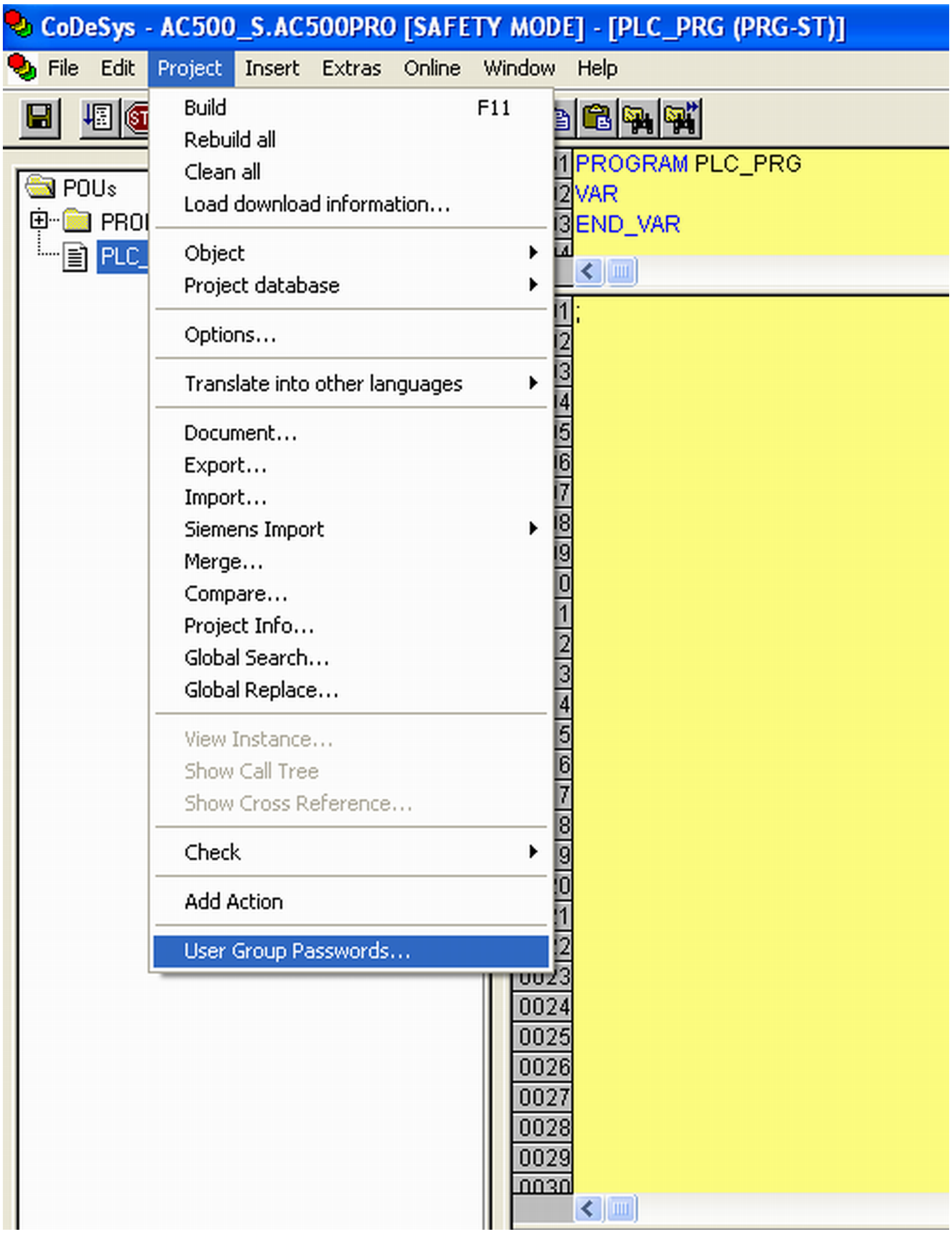


Fig. 75: Set passwords

4. Check your F-Device configuration in CODESYS Safety.

   If your configuration of F-Devices is final, you have to check that F-Parameter values from F-Parameter tab are the same as those imported to CODESYS Safety: Go to *“Resources”* tab in CODESYS Safety project. Navigate to *“Global Variables ➔ PROFIsafe”* and select the F-Device instance you want to check.

> **DANGER!**
>
> You have to formally confirm that F-Parameter values from F-Parameter tab are the same as those imported to CODESYS Safety (item 3 in *Chapter 6.2 “Checklist for creation of safety application program” on page 334*).

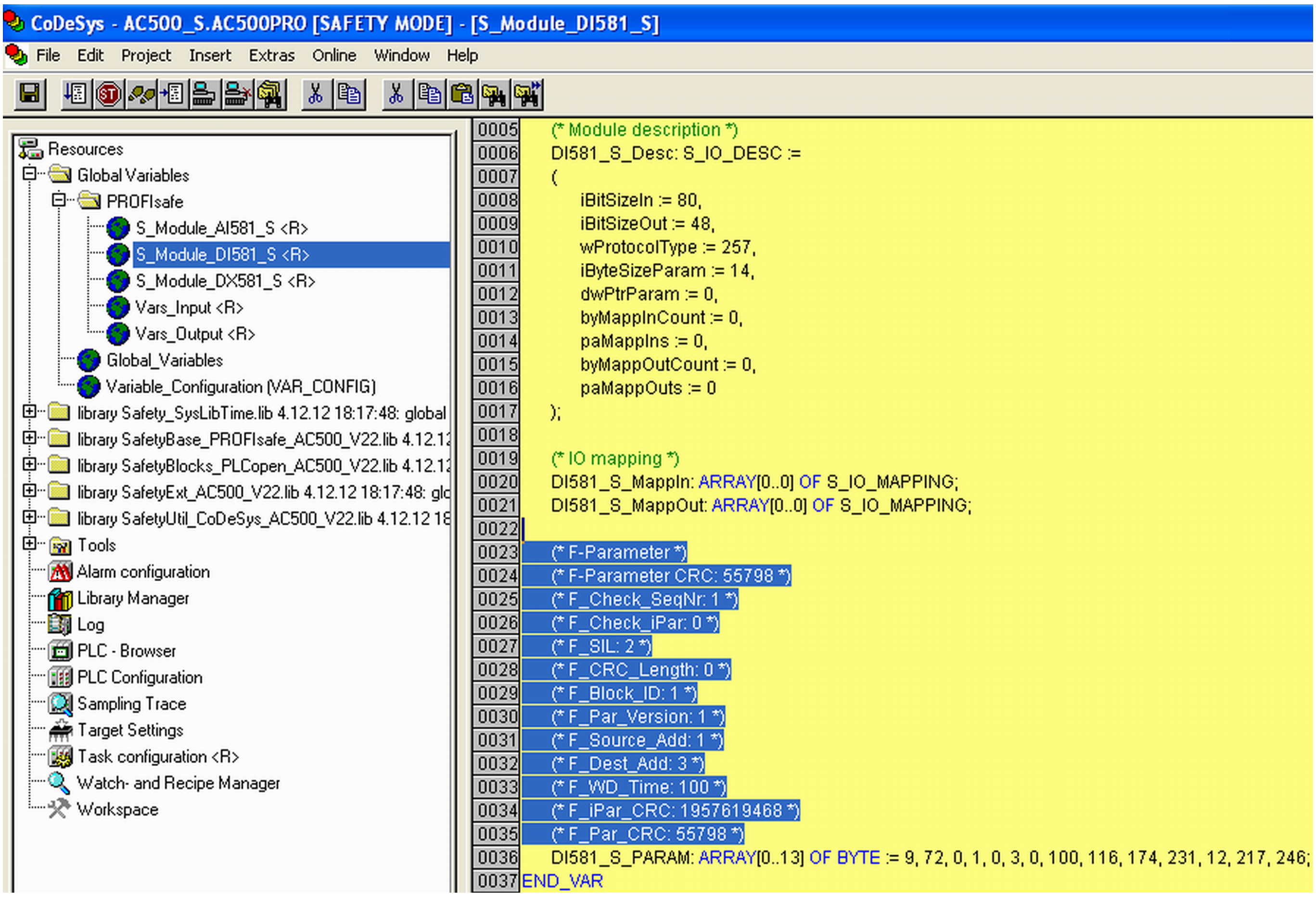


Fig. 76: F-Parameter values in CODESYS Safety

5. All configured input and output variables can be found in separate global variable lists.

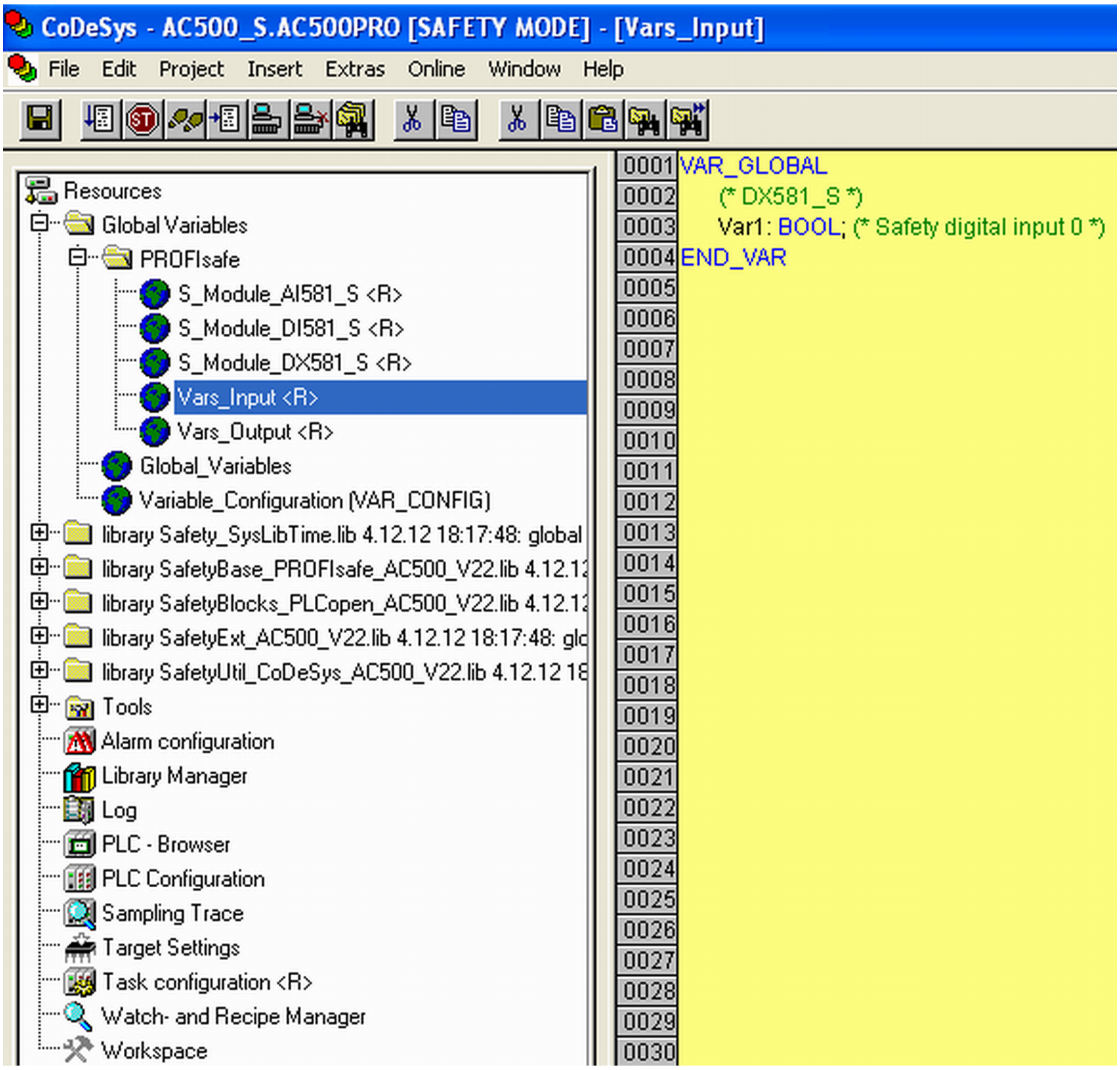


Fig. 77: Global variable list in CODESYS Safety

⇨

DANGER!

It is not allowed to change read-only (see <R> sign) resources, task configuration and pre-certified POUs (CallbackInit, CallbackReadInputs, CallbackWriteOutputs, InitPROFIsafe, ReadPROFIsafeInputs, WritePROFIsafeOutputs) under PROFIsafe folder in CODESYS Safety. A change of <R> ressources could lead to inconsistencies between Automation Builder and CODESYS Safety project.

NOTICE!

All configured safety input and output variables can also be seen in non-safety CODESYS project (e.g., for their visualization in operator panels, data logging, etc.).

The difference comparing to CODESYS Safety project is that end-user is not able to modify the values of those safety variables from non-safety CODESYS project. It is prohibited by proper design.

6. Check the validity of the safety libraries.

   In Library Manager, check that the CRCs of the used safety libraries are as listed in *Table 14 "Safety libraries" on page 189*.

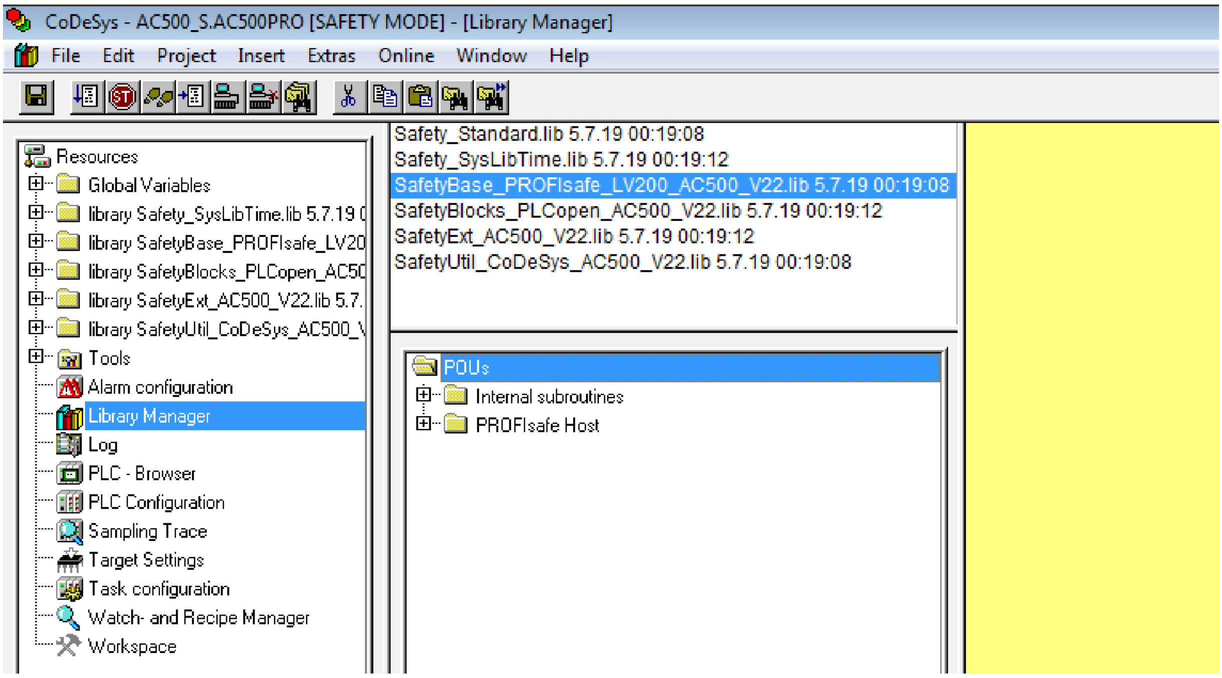


Fig. 78: All available safety libraries can be found in the Library Manager

⇨

DANGER!

The user is responsible to check that only certified safety libraries are used in his project. Refer to the overview of certified safety libraries and CRCs *Chapter 4.6.1 "Overview" on page 189*.

The user alone is responsible for all libraries which are created by him and referenced in the project for use in safety applications.

You have to formally confirm that no non-safety libraries are used in your safety application (item 19 in *Chapter 6.2 "Checklist for creation of safety application program" on page 334*).

NOTICE!

AC500-S safety CPU is a single-task machine, thus, no task configuration is needed.

7. Start programming your safety application.

The safety application program must be identified using the following properties: project name, file name, change date, title, author, version, description and CRC. Using CODESYS Safety menu item "Online/Check boot project in PLC", one can check that off-line CODESYS Safety project and the boot project on the safety CPU are identical.

Forcing of variables is supported by the safety CPU, but only in DEBUG (non-safety) mode, which means that user takes over a complete responsibility for potential damages due to wrong system behavior in the DEBUG (non-safety) mode.

> **DANGER!**
> Forcing of variables in the safety CPU is only allowed after consulting the approving board responsible for site approval. During forcing, the user in charge must ensure sufficient safety technical monitoring of the process by other technical, organizational and structural measures.

For safety applications developed with AC500-S, CODESYS visualizations using CODESYS Safety are allowed for debugging and maintenance purposes only.

> **DANGER!**
> Changing values via controls (e.g., "Write values") would cause the safety CPU to switch to a DEBUG RUN mode, which is non-safe.
>
> In case of an activation of DEBUG RUN (non-safety) mode on the safety CPU, the responsibility for safe process operation lies entirely with the organization and person responsible for the activation of DEBUG RUN (non-safety) mode.

> **NOTICE!**
> ST, FBD and LAD are the only IEC 61131 languages supported by the safety CPU for safety programming. Pay attention to CODESYS Safety programming guidelines *Chapter 4.4 “CODESYS Safety programming guidelines” on page 179*. ST with a subset defined in *Chapter 4.4* is equivalent to the limited variability language, as defined in IEC 61508.

> **NOTICE!**
> Do not create global variable lists using names beginning with the prefix "S_Module_". Global variable lists starting with "S_Module_" will be automatically updated by the CODESYS Safety and may lead to the loss of the user information.

For the safety PLC, it is important that all F-Devices are successfully initialized before program logic execution starts. F-Devices start in FV_activated mode *more details on PROFIsafe F-Host stack: Chapter 4.6.3 SafetyBase_PROFIsafe_LV200_AC500_V22.lib on page 194*. To realize a simultaneous start, we recommend using an own special POU, similar to SF_Startup explained below, which handles various possible start-up scenarios in PROFIsafe specification *[3]* and then gives "Ready" output as a trigger for further normal safety program logic execution. As you can see from the implementation below, it is enough if at least one of the channels in DI581-S module has PROFIsafe diagnostic bit set to 1, meaning that normal process values can be delivered.

Declaration part

```
FUNCTION_BLOCK SF_Startup

VAR_OUTPUT
```

```
   Ready: BOOL; (* Set to TRUE if all safety modules are
initialized *)
END_VAR

VAR
    bTempReady: BOOL; (* Set to TRUE if DI581-S safety module is
ready *)
END_VAR

VAR CONSTANT
   _TRUE: BOOL := TRUE; (* Constant because TRUE is a literal *)
    _FALSE: BOOL := FALSE; (* Constant because FALSE is a literal
*)
    wdNull: WORD := 16#0000; (* Constant for Safety I/O
initialization *)
END_VAR

VAR_EXTERNAL
    DI581_S: PROFIsafeStack; (* External declaration *)
END_VAR
```

Implementation part

```
(* Check if operator acknowledge is required for F-Device *)
IF DI581_S.OA_Req_S THEN (* The module requests an acknowledgment?
*)
   DI581_S.OA_C := DI581_S.OA_Req_S; (* Acknowledge it, if
requested *)

(* IS_DI581_Started is the input variable for all channel
PROFIsafe diagnostic bits set in Control Builder Plus / Automation
Builder for DI581-S module *)
ELSIF IS_DI581_Started > wdNull THEN (* Is this module
initialized? *)
   bTempReady := _TRUE; (* Yes, the module is initialized *)
ELSE
   bTempReady := _FALSE; (* No, the module is not initialized yet
*)
END_IF;

IF bTempReady THEN (* Set POU output signal *)
   Ready := _TRUE;
ELSE
    Ready := _FALSE;
END_IF;
```

> **NOTICE!**
> To acknowledge the F-Device after a module passivation, OA_C command bit has to be toggled from ‘0’ to ‘1’ until OA_Req_S status bit becomes "0".

8. Set up correct communication parameters.

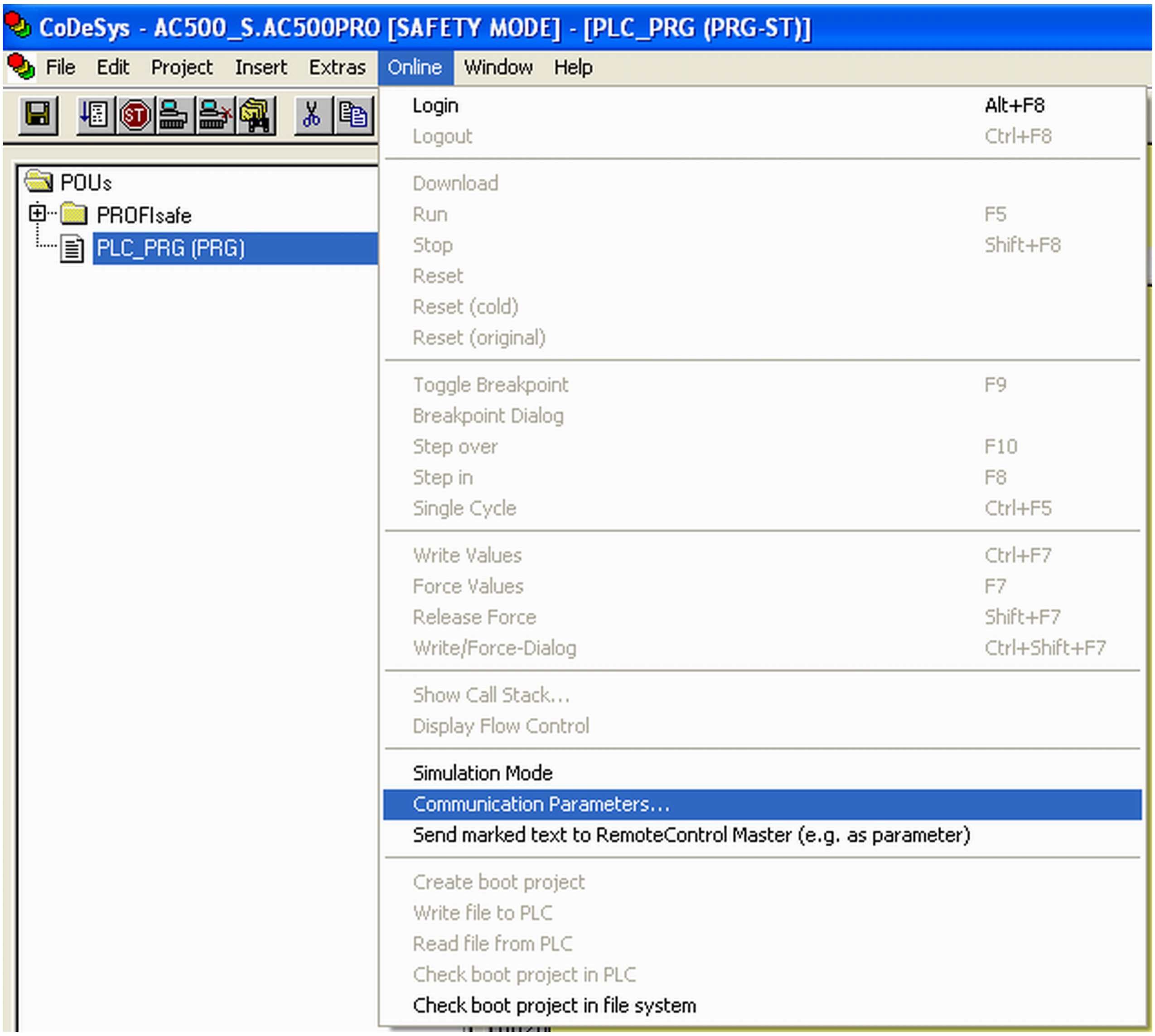


Fig. 79: Set communication parameters

⇨

> **NOTICE!**
> Make sure that to download CODESYS Safety project, either *“ABB Tcp/Ip Level 2 AC”* or *“ABB RS232 AC”* communication channels were selected.

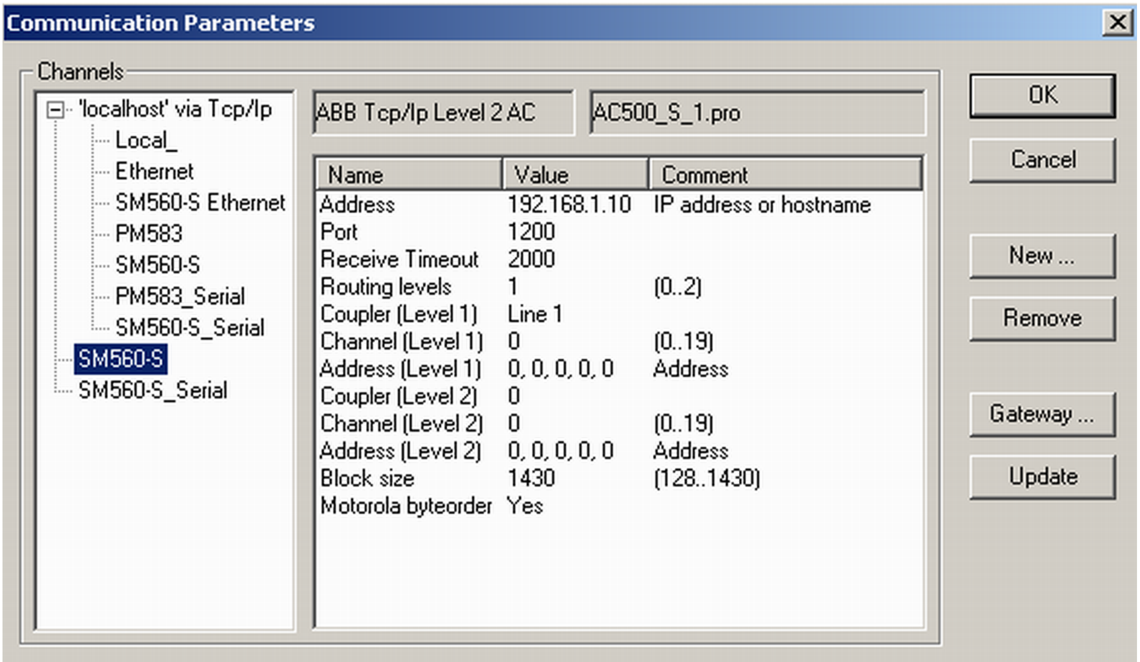


Fig. 80: Example with Ethernet connection

Note that "Address" is the IP address of your non-safety CPU, if supported on the non-safety CPU (you can also use COM port for program download using serial connection). Coupler (level 1) defines the position of the safety CPU (line 1 - position 1, line 2 - position 2 and so on).

More details on "Communication Parameters" are in ⇘ *[4]*.

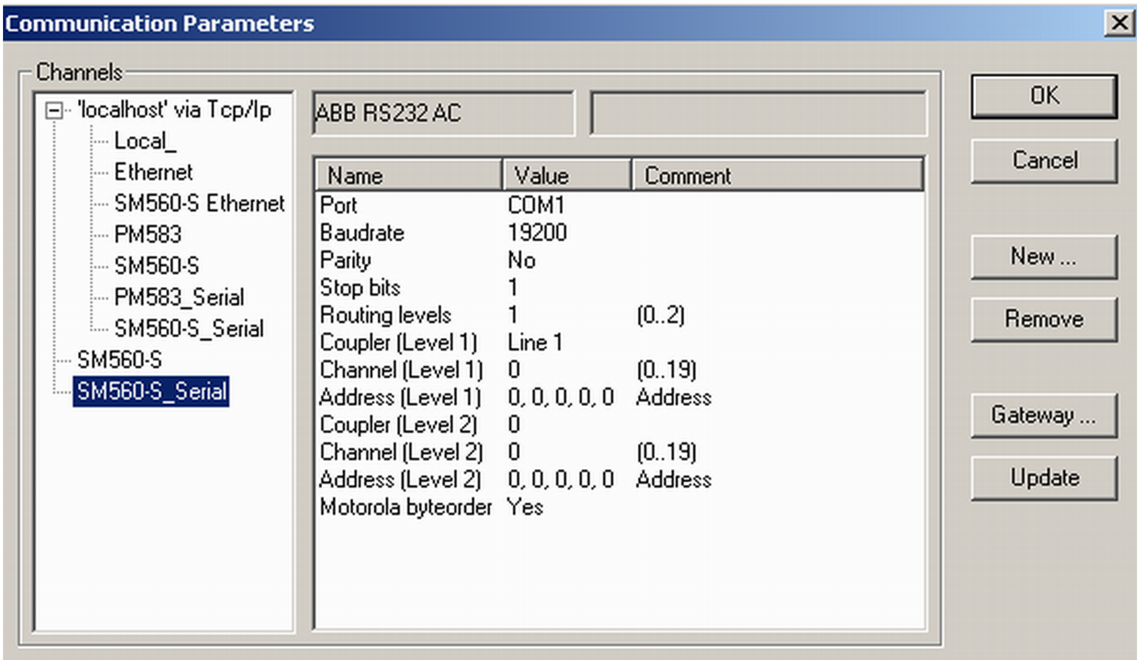


Fig. 81: Example with a serial connection

9. Download your safety application to the safety CPU.

   You can transfer your safety program to the safety CPU from a PC or using an SD card.

   ↳ *“Download your safety program to the safety CPU from a PC” on page 159*

   ↳ *“Download your safety program to the safety CPU from an SD card” on page 161*

Download your safety program to the safety CPU from a PC

10. Download your safety application and create a boot project so that your safety CPU can start safety program execution after a power cycle.

> **NOTICE!**
> The "Online Change" service of CODESYS is not supported by the safety CPU for safety reasons. It means that each program change of CODESYS Safety project requires stopping the safety CPU, downloading a new boot project and then executing a power cycle or rebooting through non-safety CPU to see the safety program change(s) become active.

> **NOTICE!**
> Only one user can be logged-on to the given safety CPU at a time. It is needed to avoid multiple changes on the safety CPU from different users working at the same time.
>
> The limitation on the number of open connections only exists for the safety CPU, which means that it is still possible to simultaneously connect to non-safety CPU, e.g., using web and OPC server functionality.

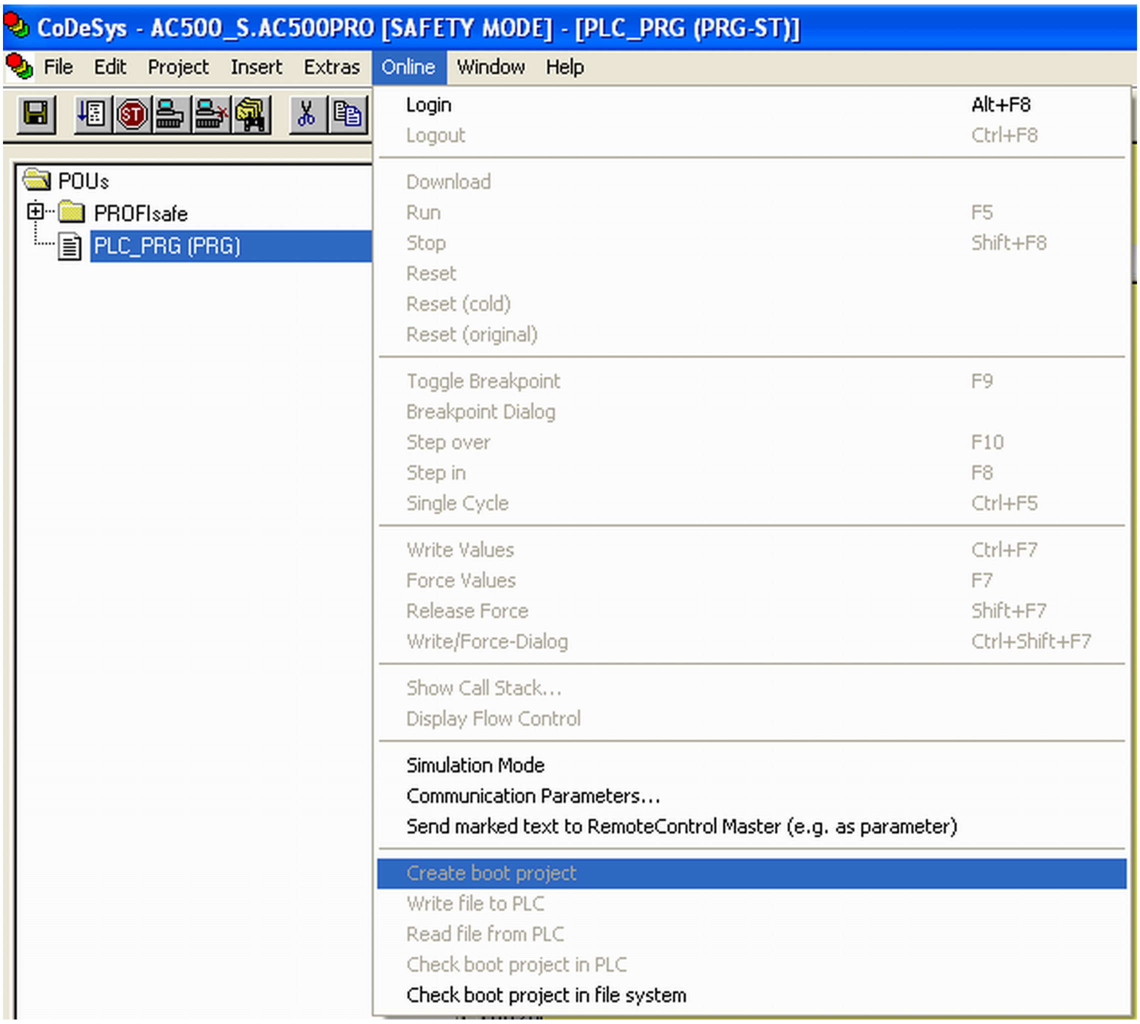


Fig. 82: Create boot project for the safety CPU

⇨

DANGER!
If *“Update Device...”* function was used on safety modules, then a full functional testing of all parts of the safety application has to be performed. This test must be carried out with the machine in its final configuration including mechanical, electrical and electronic components, sensors, actuators and software.

NOTICE!
Use CODESYS Safety menu item *“Online ➔ Check boot project in PLC”* to verify that offline CODESYS Safety project and the boot project on the safety CPU are identical (file name, change date, title, author, version, description and CRC).

The same comparison can be done with another boot project saved on the PC or SD card using *“Online ➔ Check boot project in file system”* menu item.

Note that before the boot project is created offline on the PC for a backup and later usage, the boot project has to be loaded at least once to the safety CPU.

It is highly recommended to execute *“Clean All”*, *“Rebuild All”* commands from CODESYS Safety "Project" menu before downloading the safety program to safety CPU.

NOTICE!
The boot project CRC uniquely identifies the safety CPU boot project. Note that not only code changes but also different actions in the programming environment can lead to new boot project CRC.

User actions which change the safety boot project CRC:

- In CODESYS Safety:
  - Select tab *“Resources”*, open *“Target settings”* and press *[OK]* without any changes in the dialog.
  - Select *“Project ➔ Options”* and press *[OK]* without any changes in the dialog.
  - Select tab *“Resources”*, open *“Workspace”* and press *[OK]* without any changes in the dialog.
- In Automation Builder:
  - Double-click on the safety CPU, go to the tab *“CPU Parameters Parameters”* and change any of the parameters, e.g., *“Enable debug”*. After that, open CODESYS Safety (double-click on safety application node).
  - With AC500 V2 non-safety CPU: Double-click on the safety CPU, make changes in tab *“Data exchange configuration”* and open CODESYS Safety (double-click on safety application node).

NOTICE!

Remember that non-safety CPU takes part in iParameter transfer to F-Devices, thus, you shall not only download your safety application program to safety CPU, but also in a similar way *[4]* download non-safety program from CODESYS to non-safety CPU and create a boot project for non-safety CPU.

If you do not follow the recommendation above, you may face configuration error or passivation of some F-Devices.

DANGER!

Do not use *“Write file to PLC”* command for the safety CPU because it may lead to the loss of important user information or load of corrupted data on the safety CPU.

Skip the next step and continue with the step after it.

Download your safety program to the safety CPU from an SD card

11.

DANGER!

If you transfer your safety program to safety CPU using SD card, you have to make sure that the inserted SD card contains the correct safety program. You can check this through program identification (e.g., boot project CRC) or other measures, such as a unique identifier on the SD card.

NOTICE!

The safety CPU boot project can be updated via SD card only if no boot project is present on the safety CPU *“Boot project update” on page 41*.

- Transfer the safety program to the SD card *“Boot project update” on page 41*.
- Perform a program identification - check if SD card and offline (e.g., on PC) safety program CRCs match using *“Online ➔ Check boot project in file system”* in CODESYS Safety.
- Attach an appropriate label to the SD card.

The outlined procedure must be ensured through organizational measures.

12. You can use PLC browser commands after login on safety CPU.

The following PLC browser commands (these commands can be called from CODESYS Safety) are supported by the safety CPU:

?	- List of available browser commands
reflect	- Output of browser commands (for test purposes)
pid	- It shows the project ID
pinf	- It shows project information in AC500 format
getprgprop	- It shows program properties in AC500 format
getprgstat	- It shows program status in AC500 format
setpwd	- It sets safety CPU password (it is needed during login). This command is active only if safety CPU "Enable debug" parameter was set to "ON" and proper boot project was loaded to non-safety CPU.
delpwd	- It deletes safety CPU password. This command is active only if safety CPU "Enable debug" parameter was set to "ON" and proper boot project was loaded to non-safety CPU.
rtsinfo	- It shows firmware and boot project information in AC500 format
proddata	- It shows safety CPU production data in AC500 format
diagreset	- It resets diagnosis system of the safety CPU
diagack all	- It acknowledges all errors
diagack x	- It acknowledges all errors of class x (x= 1 .. 4)
diagshow all	- It shows all errors in AC500 format
diagshow x	- It shows all errors of class x
delappl	- It deletes boot project in the flash memory. This command is executed only in DEBUG STOP state of the safety CPU. After safety CPU restart, one shall check that no boot project is available in the safety CPU. This command is active only if safety CPU "Enable debug" parameter was set to "ON" and proper boot project was loaded to non-safety CPU.
deluserdat:	- It deletes user data in the flash memory. This command is executed only in DEBUG STOP state of the safety CPU. It is executed immediately and is active only if safety CPU "Enable debug" parameter was set to "ON" and proper boot project was loaded to non-safety CPU.
applinfo	- It shows the application information, e.g., results of time profiling using functions SF_APPL_MEASURE_BEGIN and SF_APPL_MEASURE_END.
applinfo reset	- It resets all application information, e.g., time measurement values.

None of the above-mentioned safety CPU PLC browser commands changes the state (e.g., from RUN to DEBUG RUN or DEBUG STOP, etc.) of the safety CPU.

NOTICE!

The following PLC browser commands from safety CPU can influence its state:

resetprg:

It prepares safety CPU restart with initial variable values. Safety CPU changes its state, e.g., from RUN to DEBUG STOP. *This command is only accepted if safety CPU "Enable debug" parameter was set to "ON" and proper boot project was loaded to non-safety CPU.*

resetprgorg:

It restores safety CPU original state (all variables, flash memory sections, etc. get original values). Safety CPU changes its state, e.g., from RUN to DEBUG STOP. *This command is only accepted if safety CPU "Enable debug" parameter was set to "ON" and proper boot project was loaded to non-safety CPU.*

DANGER!

The results of *"delappl"*, *"resetprgorg"*, *"setpwd"* and *"delpwd"* command execution shall be checked by the end-user through a log-on with CODESYS Safety after a power cycle of the safety CPU.

4.3.6.1 Safe CPU to CPU communication using SM560-S-FD-1 and SM560-S-FD-4

SM560-S-FD-1 and SM560-S-FD-4 Safety CPUs provide up to 32 F-Device instances for safe CPU to CPU communication. The safety data of each F-Device instance is mapped to CM589-PNIO or CM589-PNIO-4 PROFINET IO device communication modules. CM589-PNIO and CM589-PNIO-4 communication modules allow physically separating their PROFINET network from that of CM579-PNIO PROFINET IO controller communication module on the same non-safety CPU.

ABB GSDML files for CM589-PNIO/CM589-PNIO-4 PROFINET devices can be used to configure process and safety data parameters in 3rd party PROFINET/PROFIsafe F-Host systems.

To support all kinds of 3rd party PROFIsafe F-Hosts, including those which limit the usage of PROFINET UseAsBits attribute in one PROFIsafe module to 64 bits, e.g., Siemens S7 3xx-F CPUs, two types of safety data descriptions were defined:

- Primary: 12 bytes defined as UseAsBits.
- Secondary (for F-Hosts which do not support 12 bytes defined as UseAsBits): 8 bytes defined as UseAsBits and two Integer16 values.

Establish a safe CPU to CPU communication using PROFINET/ PROFIsafe

1. Define master and slave controllers in the control system setup. Note that the same system could be simultaneously master and slave as well.
   - All controllers, which have to be masters only, shall have at least non-safety CPU, CM579-PNIO IO controller (if no built-in PROFINET IO controller functionality is available) and SM560-S safety CPU.
   - All controllers, which have to be slaves only, shall have at least non-safety CPU, CM589-PNIO IO device (or CM589-PNIO-4 if the communication to more than 1 PROFINET IO controller is required; usage of more than 1 CM589-PNIO communication module is also supported) and SM560-S-FD-1 Safety CPU (or SM560-S-FD-4 if the communication to more than 1 PROFINET IO controller is required).
   - All controllers, which have to be masters and slaves simultaneously, shall have at least non-safety CPU, CM579-PNIO IO controller (if no built-in PROFINET IO controller functionality is available), CM589-PNIO IO device (or CM589-PNIO-4 if the communication to more than 1 PROFINET IO controller is required; usage of more than 1 CM589-PNIO communication module is also supported) and SM560-S-FD-1 safety CPU (or SM560-S-FD-4 if the communication to more than 1 PROFINET IO controller is required).

   > **NOTICE!**
   > Only one safety CPU can be attached to the non-safety CPU. The number of PROFINET communication modules for the given non-safety CPU is only limited by the number of available slots on it.

   > **NOTICE!**
   > 3rd party PROFINET IO controllers with F-Hosts can be also used in the setup. Use CM589-PNIO / CM589-PNIO-4 GSDML files from *www.abb.com/plc* to connect AC500-S PLC as a slave to 3rd party master systems.

2. After the selection of PROFINET communication modules and safety CPUs on master and slave systems, one has to define the number of safety bytes, which have to be exchanged between the slave and master systems. Maximum 384 safety bytes can be exchanged (384 safety bytes as inputs and 384 safety bytes as outputs).

3. Safety bytes can be instantiated on slave systems by selecting, respectively, CM589-PNIO or CM589- PNIO-4 modules and instantiating "12 Byte In/Out (Safety)" or "8 Byte and 2 Int In/Out (Safety)" objects on it. The configuration of CM589-PNIO or CM589-PNIO-4 modules and instantiation of non-safety process data is explained separately in *[4]*. SM560-S-FD-1 and SM560-S-FD-4 can handle up to 32 F-Device "12 Byte In/Out (Safety)" / "8 Byte and 2 Int In/Out (Safety)" objects.

   > **NOTICE!**
   > The PROFIsafe F_Dest_Add values are consequently assigned to these instances in the Automation Builder project according to their order (no mixture is possible). The expected base address for this group is defined using the safety CPU rotary address switch and the configured F-Parameter value in the master system project *Chapter 3.1.2.5 “Address / configuration switch / F_Dest_Add settings ” on page 37*.

After the instantiation of "12 Byte In/Out (Safety)" / "8 Byte and 2 Int In/Out (Safety)" objects, one can assign variable names for instantiated IN and OUT safety data. These variables can be later used in the safety CPU application program after CODESYS Safety instance is opened. To be able to get access to the safety data in the safety CPU program, it is mandatory to give symbolic names for the required safety data.

4. In each master system configuration, one has to instantiate CM589-PNIO or CM589-PNIO-4, respectively, under CM579-PNIO to establish the PROFINET connection to slave systems ⇘ *[4]*. The PROFINET shared device functionality supported by CM589-PNIO-4 shall be also taken into account if slave system data shall be exchanged with more than one (up to 4) other control systems.
5. Similar to slave system configuration, one has to instantiate "12 Byte In/Out (Safety)" or "8 Byte and 2 Int In/Out (Safety)" objects on each master system. Note that the order of objects and their type in the master configuration must be the same as that on the slave configuration, otherwise, the configuration error can be expected in the run mode. The names of instantiated "12 Byte In/Out (Safety)" or "8 Byte and 2 Int In/Out (Safety)" objects can be freely chosen.
6. By double-clicking on each instantiated "12 Byte In/Out (Safety)" or "8 Byte and 2 Int In/Out (Safety)" object, one shall assign proper F-Parameter values. F_Dest_Add shall be set correctly for each instantiated object.

   NOTICE!
   Refer to the rules of F_Dest_Add address settings and observe that only counting upwards is allowed according to the order of modules in the Automation Builder object tree (the upper object has the lowest F_Dest_Add value) ⇘ *Chapter 3.1.2.5 “Address / configuration switch / F_Dest_Add settings ” on page 37*.

   For example, we have set the rotary address switch on the slave system safety CPU (SM560-S-FD-1 or SM560-S-FD-4) to the value of 0x01. It means that our available F_Dest_Add range is 100 ... 131 ⇘ *Chapter 3.1.2.5 “Address / configuration switch / F_Dest_Add settings ” on page 37*. The first safety object "12 Byte In/Out (Safety)" or "8 Byte and 2 Int In/Out (Safety)" must use the lowest number 100. The second must use 101 and so on.
7. As for F_Source_Add, one can use all values of the allowed range (1 - 511). One has to pay attention, however, if the slave system has also master functionality, e.g., for safety I/O modules. In the latter case, it is not allowed to use the same F_Source_Add for "12 Byte In/Out (Safety)" or "8 Byte and 2 Int In/Out (Safety)" objects as F_Source_Add used in the slave system for its own F-Devices, e.g., safety I/O modules (more details on the rules which have to be taken into account for F_Source_Add and F_Dest_Add assignment: ⇘ *Chapter 3.1.2.5 “Address / configuration switch / F_Dest_Add settings ” on page 37*).
8. After the instantiation of "12 Byte In/Out (Safety)" / "8 Byte and 2 Int In/Out (Safety)" objects in the master system configuration, one can assign variable names for instantiated IN and OUT safety data. These variables can be later used in the safety CPU application program after CODESYS Safety instance is opened. To be able to get access to the safety data in the safety CPU program, it is mandatory to give symbolic names for the required safety data. The symbolic variable names can be freely chosen, but have to be unique.
9. If SM560-S-FD-4 is used as part of PROFINET shared device communication (refer to documentation for CM589-PNIO-4 in ⇘ *[4]*) to exchange also safety data with up to 4 master systems, one has to disconnect unused safety communication modules on each master system. This allows selecting which of the configured safety communication modules ("12 Byte In/Out (Safety)" / "8 Byte and 2 Int In/Out (Safety)") in the slave system communicates to which master system. Each instantiated safety communication module can have only one connection to one of the master systems. Therefore, all safety communication modules, which are connected to other master systems, shall be set to "Disconnected" using "Disconnect module" command in the menu on the master system project. The disconnected modules will get a grey background. Using "Connect module" command in the menu for the given communication module, one can re-connect them to the given master system.

NOTICE!
If the same safety communication module is connected to more than one master system, then the connection is only established with the fastest of master systems during the start-up and parameterization phase. Other master systems do not receive any data in this case. Make sure that safety communication modules ("12 Byte In/Out (Safety)" / "8 Byte and 2 Int In/Out (Safety)") are correctly connected to master systems. Wrong configuration may result in error messages *Appendix B.2 “Error messages with AC500 V2 non-safety CPU” on page 374* *Appendix C.2 “Error messages with AC500 V3 non-safety CPUs” on page 392*.

4.3.7 Checking of program and system configuration

Check your program and system configuration. Use *Chapter 6.2 “Checklist for creation of safety application program” on page 334*.

It is important that you are able to successfully fill out the checklist and sign it. No safety program shall be approved without a positively completed checklist. If some items from the checklist cannot be fulfilled, then a proper justification shall be provided in the comment section.

4.3.7.1 Checking of program and system configuration with Safety Verification Tool (SVT)

Automation Builder 2.3.x (and newer) has an integrated Safety Verification Tool (SVT) that is installed with the AC500-S software package as a part of the Automation Builder installation.

SVT verifies the AC500-S safety configuration in Automation Builder and generates an SVT checklist that AC500-S users shall use to manually complete the functional safety verification of the Automation Builder project.

DANGER!
SVT is mandatory for use with Automation Builder 2.3.x (and newer).

In Automation Builder 2.2.x and earlier versions, there was no need to use SVT due to other procedures used to verify the functional safety integrity of the Automation Builder project.

Use SVT to verify that the CODESYS Safety project matches your safety project in Automation Builder.

4.3.7.1.1 Functionality

SVT reads the IEC 61131 program objects from the CODESYS Safety project and the description files for the safety devices in Automation Builder, verifies the data from both sources and creates the SVT checklist. The SVT checklist is a text file that you can open with any text editor and print out, if necessary. Refer to the SVT checklist examples in figures that follow.

The SVT checklist has several sections:

- A project information section with general information on the safety project *“Project information section” on page 168*.
- Sections for each safety device in the safety project *“Safety device sections” on page 168*.
- A section for the used libraries *“Libraries section” on page 172*.

SVT verifies, for example:

- The integrity of the global variables for I/O mapping for each safety device in the CODESYS Safety project.
- The integrity of the mapped I/O variables with the I/O structure description.
- The checksum of the F-Parameter for each safety device.
- The integrity of F-Parameters with F-Parameter description.

DANGER!
In addition to successfully passed automatic checks, you must successfully complete all of the manual checks in the SVT checklist.

NOTICE!
Use SVT on the final Automation Builder project after which no further changes in the functional safety project part leading to a new boot project CRC are expected.

Project information section

The SVT checklist starts with a section that is used to manually verify information regarding the whole safety project.

```
#########################################################################################################
#                                                                                                       #
# SVT (Safety Verification Tool) checklist                                                              #
#                                                                                                       #
#########################################################################################################

DANGER! You must be qualified in functional safety to do work with functional safety devices. Read and understand the
AC500-S Safety User Manual and other relevant documents before you use SVT. Refer to www.abb.com/plc.
This SVT checklist is generated by the Safety Verification Tool. Use it to verify the integrity of your safety project.
It contains the results of automatic checks done by SVT and lists the safety devices used in the project for the manual
checks. Make sure that all applicable safety devices are listed and that their data is correct. Archive this SVT
checklist for later reference, if all of the checks were successfully passed.

Generated at:     28.01.2020 10:03:52
SVT version:      1.1.0.582

The automatic checks have passed.

In the CODESYS safety project, open the "Project menu" and select "Project Info...".
[  ] The "Directory" and the "File name" are identical to the SVT checklist project directory and file name:
     C:\Users\Test\AppData\Local\Temp\CoDeSys\D5F449009B6B68336EB2211CDD25E26A__319abdfe-e0b3-428d-a4d5-8c75a600e079\
     AC500_S.AC500PRO
[  ] The "Change date" is identical to the SVT checklist project date: 28.01.2020 10:01:18
[  ] Verify using CODESYS Safety menu item "Online" / "Check boot project in PLC" that the CODESYS Safety project and
     the boot project on the safety CPU are identical and enter the boot project CRC here:
     _______________________________________________________

The "Project Info..." stored in the CODESYS safety project (for information only)
Title:           SVT project title
Author:          SVT project author
Version:         SVT project version
Description:     SVT project description

SVT data checksum:

9a9b 7a09 3624 a7f1 c9db 3a2a af20 16e5

Is the SVT data checksum identical to the SVT data checksum in the previously approved and valid SVT checklist?
[  ] Yes, the Automation Builder safety project configuration is identical. You can skip the rest of the manual checks
     and use a previously approved and valid SVT checklist with an identical SVT data checksum.
[  ] No, do the manual checks that are listed below.

Verify that all configured safety devices in the Automation Builder project are listed below and that they have a
section with the same title in this SVT checklist:
[  ] 1. DX581_S
[  ] 2. _12_Byte_In_Out_Safety
[  ] 3. SIO_02_02
[  ] All safety devices in the Automation Builder project are present in the SVT checklist.
[  ] The SVT checklist has the end indication 'End of SVT checklist', which is the last line of text.
```

Fig. 83: Example of a project information section of an SVT checklist

1 Time stamp and version information
2 Result of the automatic consistency checks done by SVT
3 Reference to the CODESYS Safety project
4 Data checksum for the whole SVT checklist
5 List of the safety devices in the safety project

Safety device sections

After the project information section, the SVT checklist has individual sections for each safety device in the safety project. The content of each safety device section depends on the type of the safety device.

ABB safety devices

```
    ##########################################################################################################################
    #                                                                                                                        #
    # 1. DX581_S                                                                                                             #
    #                                                                                                                        #
    ##########################################################################################################################

1   The automatic checks for this safety device have passed.

    Safety device data checksum:

    af76 7c66 3dfd 27be 33ca 1db6 f652 5a24

2   Is the safety device data checksum identical to the safety device data checksum in the previously approved and valid
    SVT checklist?
    [  ] Yes, the safety device configuration is identical. You can skip the manual checks for this safety device.
    [  ] No, do the manual checks that are listed below for this safety device.

3   [  ] In the Automation Builder project, select the "Information" tab on the safety device and verify that the device
         type description in the top left-hand corner is identical to the SVT device type description: DX581-S

    In the Automation Builder project, select the safety device, select the I/O mapping tab and verify that all safety
    device channels and variables are identical to the safety device channels and variables listed in this section.
    The data type and I/O (Input or Output) columns are for information only.
    ==========================================================================
    |    Variable          | Channel                            | Data type | I/O    |
    ==========================================================================
    [  ]                   | Safety digital inputs I0 - I7      |           | Input  |
    [  ] IS_Estop1         |   Safety digital input I0          | BOOL      | Input  |
    [  ]                   |   Safety digital input I1          |           | Input  |
    [  ]                   |   Safety digital input I2          |           | Input  |
4   [  ]                   |   Safety digital input I3          |           | Input  |
    [  ]                   |   Safety digital input I4          |           | Input  |
    [  ]                   |   Safety digital input I5          |           | Input  |
    [  ]                   |   Safety digital input I6          |           | Input  |
    [  ]                   |   Safety digital input I7          |           | Input  |
    -----------------------------------------------------------------------
    -
    -
    -
    -----------------------------------------------------------------------

    In the Automation Builder project, select the safety device, select the "F-Parameter" tab and verify that all
    F-Parameter values are identical to the safety device F-Parameter values listed in this section.
    =================================
    |    F-Parameter   | Value       |
    =================================
    [  ] F_Check_SeqNr | 1           |
    [  ] F_Check_iPar  | 0           |
    [  ] F_SIL         | 2           |
5   [  ] F_CRC_Length  | 0           |
    [  ] F_Block_ID    | 1           |
    [  ] F_Par_Version | 1           |
    [  ] F_Source_Add  | 1           |
    [  ] F_Dest_Add    | 2           |
    [  ] F_WD_Time     | 100         |
    [  ] F_iPar_CRC    | 1455424635  |
    [  ] F_Par_CRC     | 44422       |
    ---------------------------------
```

Fig. 84: Example of a safety device section for DX581-S safety I/O module

1 Result of the automatic consistency checks done by SVT
2 Data checksum for the safety device section
3 Safety device type description
4 Input and output mapping list for the safety device
5 List of F-Parameters for the safety device

F-Devices on AC500-S safety CPUs

F-Devices on AC500-S safety CPUs SM560-S-FD-1 and SM560-S-FD-4 include also a section with information on the position of the safety device in the safety project in Automation Builder.

```
##########################################################################################################################
#                                                                                                                        #
# 2. _12_Byte_In_Out_Safety                                                                                              #
#                                                                                                                        #
##########################################################################################################################

The automatic checks for this safety device have passed.

Safety device data checksum:

a1ad 2a14 7a8c 889c c9db 056a e782 ea48

Is the safety device data checksum identical to the safety device data checksum in the previously approved and valid
SVT checklist?
[  ] Yes, the safety device configuration is identical. You can skip the manual checks for this safety device.
[  ] No, do the manual checks that are listed below for this safety device.

[  ] In the Automation Builder project, select the "Information" tab on the safety device and verify that the device
     type description in the top left-hand corner is identical to the SVT device type description: 12 Byte In/Out
     (Safety)

[  ] In Automation Builder, verify that the position of the safety device (standard I/O modules and disconnected safety
     devices are ignored) under all CM589-PNIO(-4) node(s) is: 1

In the Automation Builder project, select the safety device, select the I/O mapping tab and verify that all safety
device channels and variables are identical to the safety device channels and variables listed in this section.
The data type and I/O (Input or Output) columns are for information only.
========================================================
|    Variable    | Channel         | Data type | I/O    |
========================================================
[  ] OS_CommOut1 | Output Byte 0   | BYTE      | Output |
[  ]             |   Bit 0         |           | Output |
[  ]             |   Bit 1         |           | Output |
[  ]             |   Bit 2         |           | Output |
[  ]             |   Bit 3         |           | Output |
[  ]             |   Bit 4         |           | Output |
[  ]             |   Bit 5         |           | Output |
[  ]             |   Bit 6         |           | Output |
[  ]             |   Bit 7         |           | Output |
--------------------------------------------------------
-
-
-
--------------------------------------------------------
```

Fig. 85: Example of a safety device section for a F-Device on AC500-S safety CPUs

1 Position of the safety device in the safety project in Automation Builder under all CM589-PNIO(-4) nodes

3rd party safety devices

3rd party safety device sections also have Module ID and information on the GSDML file in the SVT checklist.

```
########################################################################################################################
#                                                                                                                      #
# 3. SIO_02_02                                                                                                         #
#                                                                                                                      #
########################################################################################################################

The automatic checks for this safety device have passed.

Safety device data checksum:

77b8 c99c 48b1 76e9 b7e2 7caa 00d0 4fb7

Is the safety device data checksum identical to the safety device data checksum in the previously approved and valid
SVT checklist?
[  ] Yes, the safety device configuration is identical. You can skip the manual checks for this safety device.
[  ] No, do the manual checks that are listed below for this safety device.

In the Automation Builder project, select the "Information" tab on the safety device and verify that:
[  ] The device type description in the top left-hand corner is identical to the SVT device type description: SIO 02/02
[  ] The Module ID is identical to SVT module ID: sdio3-2_x1x2
Verify that the GSDML file listed in the SVT checklist is identical to the expected safety device vendor version.
[  ] C:\ProgramData\AutomationBuilder\AB_Devices_2.3\81\0x00B0_0x0100_DIM 1\SW%3D%2C HW%3D\
     GSDML-V2.3-Phoenix Contact-FL PN PN SDIO 2TX 2TX X1-X2-V1.1-20130408.xml

In the Automation Builder project, select the safety device, select the I/O mapping tab and verify that all safety
device channels and variables are identical to the safety device channels and variables listed in this section.
The data type and I/O (Input or Output) columns are for information only.
====================================================
|    Variable | Channel       | Data type | I/O    |
====================================================
[  ] IS_Byte1 | SI_1          | USINT     | Input  |
[  ]          |   Bit0        |           | Input  |
[  ]          |   Bit1        |           | Input  |
[  ]          |   Bit2        |           | Input  |
[  ]          |   Bit3        |           | Input  |
[  ]          |   Bit4        |           | Input  |
[  ]          |   Bit5        |           | Input  |
[  ]          |   Bit6        |           | Input  |
[  ]          |   Bit7        |           | Input  |
----------------------------------------------------
-
-
-
----------------------------------------------------

In the Automation Builder project, select the safety device, select the "F-Parameter" tab and verify that all
F-Parameter values are identical to the safety device F-Parameter values listed in this section.
============================
|    F-Parameter   | Value |
============================
[  ] F_SIL         | 2     |
[  ] F_Block_ID    | 0     |
[  ] F_Par_Version | 1     |
[  ] F_Source_Add  | 1     |
[  ] F_Dest_Add    | 11    |
[  ] F_WD_Time     | 150   |
[  ] F_Par_CRC     | 37844 |
----------------------------
```

Fig. 86: Example of a safety device section for a 3rd party safety device

1 Module ID
2 Information on the GSDML file

Libraries section

After the project information section and safety device sections, the SVT checklist continues with a libraries section.

```
##########################################################################################################################
#                                                                                                                        #
#  Libraries                                                                                                             #
#                                                                                                                        #
##########################################################################################################################

Verify that the library CRCs listed in this section are the same as those listed in the "AC500-S Libraries" section of
the AC500-S Safety User Manual. All other user-defined libraries and their CRCs not listed in the AC500-S Safety User
Manual must be separately verified and validated by the end-user to qualify them for the specific safety application.
================================================================
|    Library                                       | CRC Value |
================================================================
[  ] Safety_Standard.lib                           | fd5d3581  |
[  ] Safety_SysLibTime.lib                         | 672b8325  |
[  ] SafetyBase_PROFIsafe_LV200_AC500_V22.lib      | 1d881052  |
[  ] SafetyBlocks_PLCopen_AC500_V22.lib            | b6e0bc60  |
[  ] SafetyDeviceExt_LV100_PROFIsafe_AC500_V27.lib | 2eadeae9  |
[  ] SafetyExt2_LV100_AC500_V27.lib                | f3eb2fbc  |
[  ] SafetyExt_AC500_V22.lib                       | 72a88162  |
[  ] SafetyUtil_CoDeSys_AC500_V22.lib              | 6b29c54   |
[  ] SYSLIBCALLBACK.LIB                            | 62ad210d  |
[  ] Target_AC500_V22.lib                          | 8daa436   |
----------------------------------------------------------------
```

Fig. 87: Example of the libraries section

End of SVT checklist

After the libraries section, the SVT checklist ends with the line `End of SVT checklist` and, after that, optional fields like date, signature, etc.

```
---------------------------------------
End of SVT checklist
---------------------------------------

Optional fields:


  Reviewer(s):


  Machine/Application <ID>:


  Date:


  Signature:
```

Fig. 88: End of SVT checklist with optional fields

4.3.7.1.2 How to run SVT

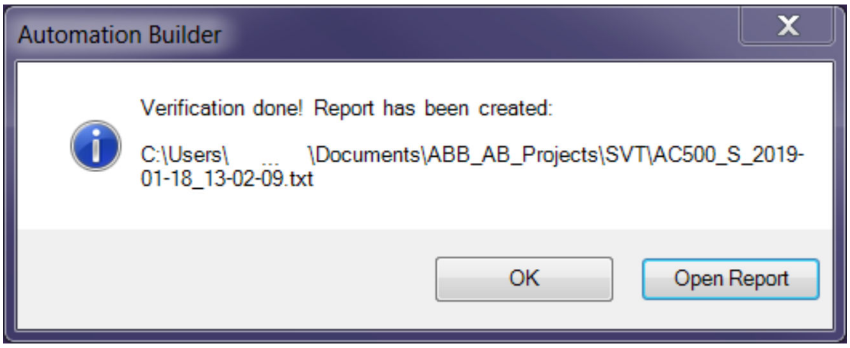
1. In Automation Builder, go to the safety CPU application node, e.g., *“AC500_S”*.
2. Right-click the node to open the context menu.
3. Select *“Create Safety Configuration Data”*.
4. Select *“Verify Safety Project Integrity”*.

> **NOTICE!**
>
> The *“Verify Safety Project Integrity”* command may not be active, e.g., when the CODESYS Safety project is open or when you did not run *“Create Safety Configuration Data”* command before.
>
> Save and close the CODESYS Safety project before you use SVT.

5. If working with password protected CODESYS Safety projects, Automation Builder will request for password *"Access to safety CPU and safety program can be protected by three passwords." on page 136*.

   ⇨ When SVT runs, the Automation Builder user interface is disabled. For large safety projects this can take several minutes.

   When the SVT run is complete, Automation Builder shows a message to indicate that the SVT verification is done.



   The message shows the path and name of the SVT checklist. The file name contains the name of the AC500-S safety CPU application node as well as the date and time of the SVT run. The date is in ISO format (YYYY-MM-DD) and the time in hours-minutes-seconds (hh-mm-ss) format.

The SVT checklist has one data checksum for the whole document file and a data checksum for each safety device in the safety project.

You can use the data checksums to verify whether the safety project has changed. If all of the data checksums in the SVT checklist are identical, there are no changes in the safety project and you do not need to repeat the manual checks.

> **NOTICE!**
> You can run SVT as often as you want to verify your safety project. We recommend that you archive the SVT checklists for final project revisions that are taken into use. You can then use the archived SVT checklists as a reference when you verify changes to your safety projects. An archived and verified SVT checklist allows you to skip sections and safety devices that you have already verified, if the data checksums did not change.
>
> You can also skip the manual checks for sections that have identical data checksums to the previous validated version of the SVT checklist. You only need to do the manual checks for sections in the SVT checklist that have a different data checksum.

For large safety projects, you can use suitable software tool to compare two textual versions of the SVT checklist to locate any differences.

4.3.7.1.3 How to verify the SVT checklist

> **NOTICE!**
> The excerpts from an SVT checklist in this section are examples only and have been edited to fit. Your SVT checklist may look different depending on the versions of Automation Builder and SVT.

Carefully read through the SVT checklist and mark the corresponding checkbox for each section and question in the SVT checklist, if the result of your verification is positive.

1. Verify the project information section ↳ *"How to verify the project information section" on page 174*.
2. Verify each safety device section ↳ *"How to verify the safety device sections" on page 175*.
3. Verify the libraries section ↳ *"How to verify the libraries section" on page 176*.
4. Verify the end of the SVT checklist ↳ *"How to verify the end of the SVT checklist" on page 176*.

If the result of your verification for at least one of manual checks in the SVT checklist is negative or not acceptable, make sure that safety configuration data is up-to-date. If the problems persist, contact ABB technical support for assistance.

Each section of the SVT checklist starts with a heading. The end of the SVT checklist is indicated with the text string:

```
--------------------------------------
End of SVT checklist
--------------------------------------
```

How to verify the project information section

This section has general information on the SVT checklist. It begins with the time stamp and the version of SVT. Example of a project information section: Fig. 83 on page 168.

1. Verify that the automatic checks done by SVT have passed:

```
The automatic checks have passed.
```

⇨ If the automatic checks generate errors, you get an error message ↳ *"Errors in the automatic checks" on page 176*:

```
The automatic checks have failed.
```

2. Verify that the CODESYS Safety project information is correct. Mark a positive verification of an item with an "X" in the SVT checklist:

```
In the CODESYS safety project, open the "Project menu" and select "Project Info...".
[ X ] The "Directory" and the "File name" are identical to the SVT checklist project directory and file name:
     C:\Users\Test\AppData\Local\Temp\CoDeSys\D5F449009B6B68336EB2211CDD25E26A__319abdfe-e0b3-428d-a4d5-8c75a600e079\
     AC500_S.AC500PRO
[   ] The "Change date" is identical to the SVT checklist project date: 28.01.2020 10:01:18
[   ] Verify using CODESYS Safety menu item "Online" / "Check boot project in PLC" that the CODESYS Safety project and
     the boot project on the safety CPU are identical and enter the boot project CRC here:
     ______________________________________________________
```

⇨

> **NOTICE!**
> Mark the corresponding checkbox for each question in the SVT checklist as in the example above. You can mark the verified items into a printout or into the text file.

3. Read the data checksum for the whole SVT checklist.

   Use this data checksum to verify changes to the entire SVT checklist.

   If you run SVT for the first time, do the following checks.

   Otherwise, check if the data checksum is identical to the previous validated SVT checklist. If these SVT data checksums are identical, you do not need to do the manual checks. If the data checksums are not identical, continue with the manual checks in the SVT checklist.

```
SVT data checksum:

9a9b 7a09 3624 a7f1 c9db 3a2a af20 16e5

Is the SVT data checksum identical to the SVT data checksum in the previously approved and valid SVT checklist?
[   ] Yes, the Automation Builder safety project configuration is identical. You can skip the rest of the manual checks
     and use a previously approved and valid SVT checklist with an identical SVT data checksum.
[   ] No, do the manual checks that are listed below.
```

4. Verify that all of the safety devices in the Automation Builder project are listed in the SVT checklist.

   If a safety device is not in the list, use *“Create Safety Configuration Data”* from the Automation Builder and run SVT again. Only configured and connected safety devices are listed in the SVT because all disconnected devices are handled outside of the given project.

```
Verify that all configured safety devices in the Automation Builder project are listed below and that they have a
section with the same title in this SVT checklist:
[  ] 1. DX581_S
[  ] 2. _12_Byte_In_Out_Safety
[  ] 3. SIO_02_02
[  ] All safety devices in the Automation Builder project are present in the SVT checklist.
[  ] The SVT checklist has the end indication 'End of SVT checklist', which is the last line of text.
```

   ⇨ After the line `End of SVT checklist` optional fields like date, signature, etc. are included.

5. Continue to verify the contents of each safety device section.

How to verify the safety device sections

Each safety device has a separate section in the SVT checklist that begins with a heading with the name of the safety device. The information in each safety device section depends on the type of the safety device *“Safety device sections” on page 168*.

1. Verify that the automatic checks done by SVT for this safety device have passed.

```
The automatic checks for this safety device have passed.
```

   ⇨ If the automatic checks generate errors, you get an error message *“Errors in the automatic checks” on page 176*:

```
The automatic checks for this safety device have failed.
```

2. Read the data checksum for the safety device.

   Use this data checksum to verify changes to the data for this safety device. If the data checksum is identical to a previously validated SVT checklist, the data for this safety device is identical and you can skip the manual checks for it. If the data checksums are not identical, repeat all of the manual checks for the safety device.

```
SVT data checksum:

9a9b 7a09 3624 a7f1 c9db 3a2a af20 16e5

Is the SVT data checksum identical to the SVT data checksum in the previously approved and valid SVT checklist?
[  ] Yes, the Automation Builder safety project configuration is identical. You can skip the rest of the manual checks
     and use a previously approved and valid SVT checklist with an identical SVT data checksum.
[  ] No, do the manual checks that are listed below.
```

3. Verify the device type description. For 3rd party devices only, also verify the Module ID.

```
In the Automation Builder project, select the "Information" tab on the safety device and verify that:
[  ] The device type description in the top left-hand corner is identical to the SVT device type description: SIO 02/02
[  ] The Module ID is identical to SVT module ID: sdio3-2_x1x2
```

4. For 3rd party devices, verify that the version of the GSDML file shown in the SVT checklist is identical to the expected version from the safety device vendor.

```
Verify that the GSDML file listed in the SVT checklist is identical to the expected safety device vendor version.
[  ] C:\ProgramData\AutomationBuilder\AB_Devices_2.3\81\0x00B0_0x0100_DIM 1\SW%3D%2C HW%3D\
     GSDML-V2.3-Phoenix Contact-FL PN PN SDIO 2TX 2TX X1-X2-V1.1-20130408.xml
```

5. If applicable, verify that the position number of the safety device in the SVT checklist corresponds to its location in the safety project in Automation Builder. The position number for the given safety device can change if their CM589-PNIO(-4) nodes are moved in the project.

```
[  ] In Automation Builder, verify that the position of the safety device (standard I/O modules and disconnected safety
     devices are ignored) under all CM589-PNIO(-4) node(s) is: 1
```

6. Verify the I/O mapping information for the safety device.

   Note that *“Data type”* and *“I/O”* are listed for information only.

```
In the Automation Builder project, select the safety device, select the I/O mapping tab and verify that all safety
device channels and variables are identical to the safety device channels and variables listed in this section.
The data type and I/O (Input or Output) columns are for information only.
=================================================================================
|    Variable          | Channel                            | Data type | I/O    |
=================================================================================
[  ]                   | Safety digital inputs I0 - I7      |           | Input  |
[  ] IS_Estop1         |   Safety digital input I0          | BOOL      | Input  |
[  ]                   |   Safety digital input I1          |           | Input  |
[  ]                   |   Safety digital input I2          |           | Input  |
[  ]                   |   Safety digital input I3          |           | Input  |
[  ]                   |   Safety digital input I4          |           | Input  |
[  ]                   |   Safety digital input I5          |           | Input  |
[  ]                   |   Safety digital input I6          |           | Input  |
[  ]                   |   Safety digital input I7          |           | Input  |
---------------------------------------------------------------------------------
-
-
-
------------------------------------------------------------------------
```

7. Verify F-Parameter values for the safety device.

```
In the Automation Builder project, select the safety device, select the "F-Parameter" tab and verify that all
F-Parameter values are identical to the safety device F-Parameter values listed in this section.
=================================
|    F-Parameter   | Value      |
=================================
[  ] F_Check_SeqNr | 1          |
[  ] F_Check_iPar  | 0          |
[  ] F_SIL         | 2          |
[  ] F_CRC_Length  | 0          |
[  ] F_Block_ID    | 1          |
[  ] F_Par_Version | 1          |
[  ] F_Source_Add  | 1          |
[  ] F_Dest_Add    | 2          |
[  ] F_WD_Time     | 100        |
[  ] F_iPar_CRC    | 1455424635 |
[  ] F_Par_CRC     | 44422      |
---------------------------------
```

Do these manual checks for each safety device in the SVT checklist. You can skip the sections for safety devices only if the data checksum for the safety device is identical to the previously validated and approved SVT checklist.

How to verify the libraries section

In the libraries section (Fig. 87 on page 172), verify that the library CRCs correspond to the AC500-S libraries ↻ *Chapter 4.6 “AC500-S libraries” on page 189*.

How to verify the end of the SVT checklist

Verify that the SVT checklist ends with the line *“End of SVT checklist”*, and if so, mark the corresponding checkbox in the project information section (Fig. 88 on page 172).

```
[  ] The SVT checklist has the end indication 'End of SVT checklist', which is the last line of text.
```

Errors in the automatic checks

If there are errors in the automatic consistency checks, SVT shows this with an error message in the project information section of the SVT checklist.

```
####################################################################################################
#                                                                                                  #
# SVT (Safety Verification Tool) checklist                                                         #
#                                                                                                  #
####################################################################################################

DANGER! You must be qualified in functional safety to do work with functional safety devices. Read and understand the
AC500-S Safety User Manual and other relevant documents before you use SVT. Refer to www.abb.com/plc.
This SVT checklist is generated by the Safety Verification Tool. Use it to verify the integrity of your safety project.
It contains the results of automatic checks done by SVT and lists the safety devices used in the project for the manual
checks. Make sure that all applicable safety devices are listed and that their data is correct. Archive this SVT
checklist for later reference, if all of the checks were successfully passed.

Generated at:     28.01.2020 10:05:23
SVT version:      1.1.0.582

The automatic checks have failed.
  - Internal error in the safety device 3. 'SIO_02_02'. For error codes, refer to the section for this safety device.

Remedy:
  - Reinstall the GSDML file of safety device 3. 'SIO_02_02' and update this device in Automation Builder.
  - Repeat the "Create Safety Configuration Data" command for your CODESYS Safety project in Automation Builder.

If the error persists, contact ABB technical support.

In the CODESYS safety project, open the "Project menu" and select "Project Info...".
[  ] The "Directory" and the "File name" are identical to the SVT checklist project directory and file name:
     C:\Users\Test\AppData\Local\Temp\CoDeSys\D5F449009B6B68336EB2211CDD25E26A__319abdfe-e0b3-428d-a4d5-8c75a600e079\
     AC500_S.AC500PRO
[  ] The "Change date" is identical to the SVT checklist project date: 28.01.2020 10:01:18
[  ] Verify using CODESYS Safety menu item "Online" / "Check boot project in PLC" that the CODESYS Safety project and
     the boot project on the safety CPU are identical and enter the boot project CRC here:
     _______________________________________________

The "Project Info..." stored in the CODESYS safety project (for information only)
Title:          SVT project title
Author:         SVT project author
Version:        SVT project version
Description:    SVT project description

SVT data checksum:

Not available

Verify that all configured safety devices in the Automation Builder project are listed below and that they have a
section with the same title in this SVT checklist:
[  ] 1. DX581_S
[  ] 2. _12_Byte_In_Out_Safety
[  ] 3. SIO_02_02                                                                                ERROR
[  ] All safety devices in the Automation Builder project are present in the SVT checklist.
[  ] The SVT checklist has the end indication 'End of SVT checklist', which is the last line of text.
```

Fig. 89: Example of an SVT checklist with errors. When there are errors in the automatic consistency checks, the contents of the project information section of the SVT checklist is slightly different.

1 List of errors encountered by the automatic consistency checks done by SVT
2 List of remedies suggested by SVT to correct the causes of errors
3 No data checksum is given for the SVT checklist when there are errors
4 List of the safety devices indicates which safety devices have generated errors

> **NOTICE!**
> If you cannot remedy all of encountered errors with the suggested remedies or otherwise, contact ABB technical support for assistance.

In addition to the project information section, each safety device section with errors has a corresponding message.

```
####################################################################################################
#                                                                                                  #
# 3. SIO_02_02                                                                                     #
#                                                                                                  #
####################################################################################################

1  The automatic checks for this safety device have failed.
     - Error in the GSDML file. Error code 6602.

2  Safety device data checksum:

   Not available

3  Device identification:
   Name                 SIO_02_02
   Device Type Desc.    SIO 02/02
   GSDML                C:\ProgramData\AutomationBuilder\AB_Devices_2.3\81\0x00B0_0x0100_DIM 1\SW%3D%2C HW%3D\
                        GSDML-V2.3-Phoenix Contact-FL PN PN SDIO 2TX 2TX X1-X2-V1.1-20130408.xml
   Device Type          81
   Device ID            0x00B0_0x0100_DIM 1
   Device Version       SW=, HW=
   Module ID            sdio3-2_x1x2
```

Fig. 90: Example of a safety device section with errors. When there are errors in the automatic checks for a safety device, the contents of the safety device section of the SVT checklist is slightly different.

1 List of errors for this safety device with exemplary error codes
2 No data checksum is given for the safety device when there are errors
3 Device identification information to help in troubleshooting the issue

Summary of error messages

Possible errors generated by SVT:

- General errors:
  – Internal error in the safety device N. 'XYZ'. For error codes, refer to the section for this safety device.
  – Internal error in CODESYS Safety project. Error code x.
  – Maximum number of 32 connected F-Devices has been exceeded.
- Errors related to safety devices:
  – Internal error in the safety device. For error codes, refer to the section for this safety device.
  – Internal error in the safety device. Error code x.
  – Internal error in the safety device or GSDML file. For error codes, refer to the section for this safety device.
  – Internal error in the safety device or GSDML file. Error code x.
  – Internal error in F-Parameters. For error codes, refer to the section for this safety device.
  – Internal error in F-Parameters. Error code x.
  – Error in the GSDML file. Error code x.
  – Missing GSDML file.
- Errors related to F-Parameters or channel mapping:
  – Internal error. Error code x.
  – Multiple mappings to an output are not permitted. Use either the parent element or sub-elements.

Possible remedies suggested by SVT to correct errors:

- Reinstall the GSDML file of safety device N. 'XYZ' and update this device in Automation Builder.
- Use either the parent element or sub-elements in the safety device N. 'XYZ'
- Repeat the *"Create Safety Configuration Data"* command for your CODESYS Safety project in Automation Builder.
- Delete or disconnect F-Devices in order not to exceed the maximum number of 32.

4.4 CODESYS Safety programming guidelines

This chapter and sub-chapters present an extract of AC500-S safety CPU relevant rules from CODESYS V2.3.x safety guidelines ➪ *[1]*.

4.4.1 Overview

CODESYS is usually used for creating non-safety applications. CODESYS is also suitable for creating safety applications of certain classes if it is used in a suitable environment in conjunction with controllers like AC500-S, specially approved for this purpose. However, this requires certain guidelines to be followed, which are described in this document.

4.4.1.1 Target group

This document is aimed at users who wish to create safety applications with CODESYS.

It also serves as a basis for testers who approve safety applications.

4.4.1.2 Requirements

To understand this document, knowledge of IEC 61131-3 ➪ *[5]*, particularly the CODESYS programming system, is required.

Experience with the creation of safety applications is helpful.

4.4.1.3 Terms

Output - Variable that is mapped to an IEC output address (%Q)

Output parameter - VAR_OUTPUT of a program or function block

Inputs - Variable that is mapped to an IEC input address (%I)

Input parameter - VAR_INPUT of a program, function or function block

4.4.2 Framework

4.4.2.1 Safety integrity level (SIL)

CODESYS is suitable for creating applications up to SIL 3. The use of CODESYS is not permitted for higher levels.

4.4.2.2 Approved CODESYS version

The following product component versions are approved for creating safety applications:

Type of product component	Name of product component	Version (date)
Programming system	CODESYS	2.3.9.9 or higher

The CODESYS version can be checked via *“Help ➔ About”*. The correct version of the runtime system is indicated by SIL 3 approval of the control system through the German Technical Inspection Association (TÜV SÜD).

4.4.2.3 Control-specific application notes

Safety controllers require a special procedure for loading safety applications. In CODESYS, the download of the bootproject is considered as safe, as it is secured by the appropriate mechanisms.

Procedure in CODESYS for loading safety application

1. Compile the user application.
2. Connect to the controller. This is secured by password protection. It causes automatic compile of user application, if needed.
3. Execute menu item *"Online ➔ Create Boot Project"*.
4. Reboot the controller.

   ⇨ It causes loading and starting of the application.

All online commands like the following disable the safe operation:

- Download
- Online change
- Set breakpoint
- Write values
- Force values
- Trace
- Single cycle
- Start/Stop
- Flow control

The variable monitoring in online mode does not disable the safe operation.

4.4.2.4 Application creation procedure

Application creation must follow the guidelines of relevant safety standards, e.g., IEC 61508 for functional safety and ISO 13849-1 for machinery. In addition to comprehensive documentation of requirements, architecture and module interfaces, this also includes full functional testing of all parts of the safety application. This test must be carried out with the machine in its final configuration including mechanical, electrical and electronic components, sensors, actuators, and software. Testing in a special test environment, for example using a debugger, may facilitate passing the final test, but cannot be used as a substitute.

4.4.2.5 Settings

Table 10: The following system settings are required:

Setting	Value
Replace constants	Selected in *"Project ➔ Options ➔ Build"*
Actions hide programs	Selected in *"Project ➔ Options ➔ Build"*

4.4.2.6 Classification

In principle most language constructs can be used in safety applications. However, for some constructs that are associated with an increased fault potential during application creation this is only possible to a limited extent and compliance with additional fault prevention measures is strongly recommended. These measures are listed with the respective construct.

4.4.3 Language-specific programming guidelines

4.4.3.1 Safety-related restrictions for developers

There are some restrictions to developing safety applications with CODESYS which have to be secured by organizational means. These are as follows:

- For safety applications, CODESYS visualizations are allowed for displaying purposes only. Changing values via controls (e.g., "Write values" *Chapter 4.4.2.3 “Control-specific application notes” on page 180*) would cause the runtime system to switch into non-safe mode without necessarily telling the user.

4.4.3.2 Language

Of the IEC 61131-3 languages implemented in CODESYS, "Structured Text" (ST), "Function Block Diagram" (FBD) and "Ladder Logic Diagram" (LAD) are approved for creating safety applications.

4.4.3.3 Task system

Due to poor testability it is only advisable to a limited extend to use multitasking for safety applications. For application with CODESYS this means:

- The complete application consisting of safety parts and non-safety parts should be called from program "PLC_PRG". To achieve a well arranged structure of the program, no logic processing should be programmed in "PLC_PRG". Assignments, calls to programs, function blocks or functions are allowed.
- The controller-specific options for monitoring total execution time must be activated and set significantly below the fault tolerance time.

4.4.3.4 Variable declarations

Of the variable types and attributes defined in IEC 61131-3 the following are suitable for creating safety applications:

Keyword	Description	Suitable (yes / to a limited extent / no) (comment)
VAR	Local block variable	Yes
VAR_INPUT	Block input parameter	Yes
VAR_OUTPUT	Block output parameter	Yes
VAR_IN_OUT	Block reference parameter	To a limited extent. (To illustrate the side effect the parameter should be identified with a prefix. Even better would be to use an input and output parameter instead.)
VAR_GLOBAL	Global variable	Yes. (We strongly recommend identifying global variables with a prefix such as "G_" or "GS_" (for safety variables).)
VAR_EXTERNAL	Declaration of global variables used in the block	Yes/optional (not required in CODESYS)
AT	Variable address allocation	Yes *Chapter 4.4.3.5 “Direct addresses” on page 182*
CONSTANT	Declaration as constant (no write access possible)	Yes. (We recommend to declare each constant explicitly.)

Keyword	Description	Suitable (yes / to a limited extent / no) (comment)
RETAIN	Variable value is preserved after switch-off	No, not supported
PERSISTENT	Variable value is preserved after reloading	No, not supported

In the interest of better readability the following rules should be followed for the declaration of variables:

- Only one block of declaration type (e.g., VAR, VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR_GLOBAL and combinations with CONSTANT) per component
- Only one variable declaration per line with informative comment

  Bad:

  ```
  VAR
     A, B, C: BOOL; (* several variables *)
  END_VAR
  ```

  Good:

  ```
  VAR
     A: BOOL; (* first variable *)
     B: BOOL; (* second variable *)
     C: BOOL; (* third variable *)
  END_VAR
  ```
- Local variables (VAR) should always have a different name. Obscuring of global variables through local variables must be avoided.

4.4.3.5 Direct addresses

The following rules must be followed when using addresses for creating safety applications:

- No application of addresses directly in the program code. Each used address must be assigned to a variable with "AT" in the declaration. In addition, we recommend identifying input/output variables through a prefix and defining them together in a single variable list.
- The application of marker addresses (%M) should be limited to a minimum due to the error-proneness of the allocation and the lack of purpose (memory for variables is allocated automatically).
- Multiple address allocation should be avoided due to obscure side effects. For word- and bit-wise access a variable is defined for the word and accessed via bit access <variable>.<bit number>.
- No address declarations within programs, function blocks, functions and data structures.

4.4.3.6 Data types

Of the data types defined in CODESYS the following are approved for creating safety applications:

Table 11: Simple data type

Keyword	Suitable (yes / to a limited extent / no) (comment)
BOOL	Yes
BYTE, SINT, USINT	Yes
WORD, INT, UINT	Yes
DWORD, DINT, UDINT	Yes
TIME, TOD, DATE, DT	Yes

Keyword	Suitable (yes / to a limited extent / no) (comment)
STRING	To a limited extent. (Technically possible, although it makes little sense due to the lack of safety input/output devices.)
REAL	To a limited extent. (Prone to error through rounding errors, therefore no query with EQ operator; check for invalid operations such as division by zero, square root of a negative number, logarithm of a negative number.)

Table 12: Complex data types

Keyword	Suitable (yes / to a limited extent / no) (comment)
ARRAY	To a limited extent. (Only with explicit range check, otherwise too prone to errors.)
STRUCT	Yes
Listing types	Yes
Subrange types	Yes
POINTER	To a limited extent. (Recommended measures: no pointer arithmetic, range check, new allocation of pointer value at the start of each cycle.)

The following rules must be followed when complex data types are used:

- For complex data types we recommend using type declarations.
- Before each access to an array an explicit range check of the index should be carried out. In the event of a violation that cannot be explained through the application, the control system should be switched to a safe state.

DANGER!
The memory access using POINTERs (e.g., ADR function) is error-prone and is generally not recommended. If used in safety applications, then the responsibility for correct usage of these and related functions lies entirely with the organization and persons who use those functions in AC500-S safety PLC.

4.4.3.7 Blocks

All IEC 61131-3 block types are suitable for creating safety applications:

- PROGRAM
- FUNCTION
- FUNCTION_BLOCK

If blocks are used, the following programming guidelines should be followed:

- Functions and function blocks must not affect global application states. This can be achieved through write access to global data and by calling system components.
- Explicit parameter transfer is preferable for calling programs and function blocks.
  Bad:
  ```
  Inst.Param1 := 7;
  Inst.Param2 := 3;
  Inst();
  X := (Inst.Out1 AND A) OR B;
  ```
  Good:
  ```
  Inst(Param1 := 7, Param2 := 3, Out => Result);
  X := (Result AND A) OR B;
  ```
- All input parameters should be assigned for a call.

4.4.3.8 Libraries

External libraries approved by the manufacturer of the control system (i.e. implemented in the firmware of the control system) may be used for safety applications.

Of the standard CODESYS libraries only the following are approved:

Library	Description	Version (date)
Safety_Standard.lib (former Standard.lib)	Standard IEC 61131-3 functions: ● Timer ● Counter ● Trigger ● Flip-flops ● String processing	2.3 (04.10.2005) or higher

User libraries created by the manufacturer of the control system or the end user may be used. On insert of a library, it has to be checked whether the selected library was actually inserted. The respective information is shown when the library is inserted.

4.4.3.9 Expressions

4.4.3.9.1 General

The following general rules must be followed for programming of expressions in safety applications:

- Mixing of different data types in an expression should be avoided. If mixing is unavoidable explicit type conversion should be used instead.
- The complexity of expressions should be minimized through the following measures:
  - Limitation of nesting depth (e.g., no more than 3 nesting levels) per expression.
  - No more than 10 operators and 10 operands per expression.
  - No application of expressions in array indices of array access.
  - No application of expressions in function parameters, function block parameters or program parameters.

4.4.3.9.2 Constants

In the interest of more transparent semantics constants should either be declared explicitly or associated with explicit typification.

Bad:

```
VAR
    size: REAL;
    diameter: REAL;
END_VAR
size:= diameter * 3.14;
```

Good:

```
VAR CONSTANT
     PI: REAL := 3.14;
END_VAR
VAR
     size: REAL;
     diameter: REAL;
```

```
END_VAR
size:= diameter * PI;
```

Also good:

```
VAR
   size: REAL;
   diameter: REAL;
END_VAR
size:= diameter * REAL#3.14;
```

4.4.3.9.3 Assignments

If assignments are used, the following programming guidelines should be followed:

- For each instruction only one assignment is permitted. The expression assignments possible in CODESYS must not be used for safety applications.
  Bad:
  ```
  Res1 := Res2 := FunCall(1, C := D, 3);
  ```
  Good:
  ```
  C := D;
  Res2 := FunCall(1, C, 3);
  Res1 := Res2;
  ```
- The implicit conversion between unsigned, signed and bit string types realized in CODESYS and the extension of smaller types to larger types during assignment should not be used. Explicit conversion should be used instead.

4.4.3.9.4 Parentheses

Through definition of priorities for operators each expression is uniquely defined even without parentheses. However, in order to avoid mistakes and improve readability the use of parenthesis is highly recommended except in very familiar cases (multiplication/division before addition/subtraction).

Bad:

```
X := A < B AND NOT A > C + D OR E;
```

Good:

```
X := (A < B) AND NOT(A > (C + D)) OR E;
```

4.4.3.9.5 Bit access

Bit access (<variable>.<bit number>) is approved for creating safety applications and should also be used instead of the regularly used multiple address allocation.

Bad:

```
VAR_GLOBAL
   Flags AT %QW12: WORD;
   Enable AT %QX12.0: BOOL;
END_VAR
Flags := 0;
Enable := TRUE;
```

Good:

```
VAR CONSTANT
    EnableBit: INT := 0;
END_VAR
VAR
    Flags AT %QW12: WORD;
END_VAR
Flags := 0;
Flags.EnableBit := TRUE;
```

4.4.3.9.6 Conversions

No implicit type conversions should be used for assignment and mixed types, i.e., only explicit conversions should be used.

Bad:

```
VAR
    A: BYTE;
    B: INT;
    C: DWORD;
END_VAR
C := A + B;
```

Good:

```
VAR
    A: BYTE;
    B: INT;
    C: DWORD;
END_VAR
C := INT_TO_DWORD(B + BYTE_TO_INT(A));
```

An even better solution in such cases is to reflect on type allocation.

4.4.3.10 Operators

The following table indicates the suitability of operators for creating safety applications.

Keyword	**Suitable (yes / to a limited extent / no) (comment)**
AND, OR, NOT, XOR	Yes
+, -, *, /, MOD	Yes. (Division should include an explicit test for divisor <> 0.)
=, <>, >, >=, <, <=	Yes
SQRT, SIN, COS, TAN, ASIN, ACOS, ATAN, LOG, LN, EXPT, EXP	To a limited extent. (Prone to error through rounding errors.)
MIN, MAX, LIMIT	Yes
MUX, SEL	Yes. (Please note: branches that are not selected are not executed. This can lead to problems if functions calling system libraries are used.)

Keyword	Suitable (yes / to a limited extent / no) (comment)
TIME	Yes
ADR	To a limited extent. (Required for POINTERS that may be used to a limited extent.)
INDEXOF	To a limited extent. (Only used as parameter for runtime system functions. The function used should be treated like an independent task.)
SIZEOF	Yes
ROL, ROR, SHR, SHL	Yes

4.4.3.11 Language constructs

The following ST language control elements are suitable for creating safety applications:

Keyword	Suitable (yes / to a limited extent / no) (comment)
IF	Yes
CASE	Yes
FOR	Yes
WHILE	To a limited extent. (Proof of avoidance of an infinite loop is required.)
REPEAT	To a limited extent. (Proof of avoidance of an infinite loop is required.)
EXIT	To a limited extent. (Exits a loop immediately. A loop should only be exited through its end condition leave.)
RETURN	To a limited extent. (Exits a subroutine immediately. A subroutine should only be exited once all instructions have been processed.)

4.4.4 General programming guidelines

In addition to language-specific guidelines, errors should be avoided through compliance with additional general guidelines. These guidelines are listed here in no particular order:

- Few states
  States in the form of variables that retain their value beyond a control cycle hamper the testability of an application. This can be avoided with the following measures:
  – Avoidance of states wherever possible
  – A state variable should only be described once per cycle. This facilitates tracing of errors if a state has an invalid value.
  – If a state consists of several variables it should be encapsulated in a function block. State transitions should only be affected by calling the block.
- No warnings
  A safety application must not generate compiler warnings!
- Limited number of rows (500) per block
  In the interest of transparency, a block should have no more than 500 rows.
- Limited number of characters per row (150)
  In the interest of transparency, a row should have no more than 150 characters
- No re-use of variables
  Each variable should only be used for one purpose. Application in another context, even if the previous purpose is no longer important, involves a significant fault potential, particularly for modifications.
- Variables as local as necessary
  Variables that are only described in one block must be declared locally. The only exception is variables that are linked with addresses. These should be declared globally in order to avoid multiple assignments.

- Only one access to output
  As for states, outputs should only be described at one point in the program.
- No access to global variables from functions and function blocks
  A function should have no side effects, a function block should only change the state of its own instance. Functions and function blocks should therefore not access global variables.

4.4.5 Safety and non-safety parts of the application

For very complex applications, it is advisable to transfer all safety application parts to a separate control system. If this is not possible, the application parts should be separated through the following measures:

- Blocks (programs, function blocks and functions) are either safety blocks or not. All safety blocks should be identified through a prefix (e.g., "S_").
- Calls of non-safety blocks in safety blocks are not permitted. This must be checked with the "Show project call tree" function.
- Calls of safety blocks in non-safety blocks are limited to standard functions. This must be checked with the "Show project call tree" function.
- Global variables are either safety or not. All safety variables should be identified through a prefix (e.g., "S_"). All safety variables are defined in separate variable lists that are also identified through a prefix.
- Write access to safety variables from non-safety blocks is not permitted. This must be checked with the "Show project cross-reference list" function.
- Write access to non-safety variables from safety blocks is not permitted. This must be checked with the "Show project cross-reference list" function.
- The I/O addresses for safety application parts and non-safety application parts are separated into different ranges, while the safety parts coming first (lower addresses) in the memory.
- The following measures should also be adhered to in the non-safety part:
  – Limited application of pointers
  – Range check of indices before write access to fields (ARRAY)
  – No multiple address allocation

4.5 Safety code analysis tool

Instead of manually checking CODESYS Safety programming guidelines, one can use ABB software tool "AC500-S Safety Code Analysis" (SCA) to automatically check most of the safety rules defined by CODESYS.

The detailed description on how to use ABB SCA tool can be found at www.abb.com/plc and in its help system. AC500-S SCA tool can be downloaded for free from www.abb.com/plc.

There are rules which still have to be checked manually ⇨ *Table 13 “CODESYS Safety programming rules to be checked manually” on page 188*. AC500-S SCA tool is not able to detect them in the safety application program.

Table 13: CODESYS Safety programming rules to be checked manually

Rule for manual check in CODESYS Safety	Comments (relevance for AC500-S)
Verify that the watchdog is activated. Verify that the watchdog time is set sufficiently shorter than the process failure response time.	Use a special library POU SF_WDOG_TIME_SET ⇨ *Chapter 4.6.7.3 “SF_WDOG_TIME_SET” on page 309*
Verify that there is only one task.	AC500-S supports only one task, thus, there is no need for this check.
Verify that, other than standard libraries, only libraries certified for safety applications are used.	These rules are included in ⇨ *Chapter 6.2 “Checklist for creation of safety application program” on page 334*
For each POU, verify that there are no unnecessary state variables.	

Rule for manual check in CODESYS Safety	Comments (relevance for AC500-S)
Verify that the following holds for all function blocks: If more than one variable is used to store state information, encapsulate these variables into their own function block and only use calls on this function block to change the state.	
Verify that the compiler reports neither errors nor warnings when compiling the application.	
For each POU, verify that variables are not re-used later on with a different meaning.	
Verify that the names of safety POUs start with "S_". Verify that the names of non-safety POUs do not start with "S_".	These rules have to be checked only if you plan to implement not only safety but also non-safety functions on AC500-S safety CPU. In typical applications with AC500-S it is not the case, because non-safety functions are realized on non-safety CPUs.
Verify that names of safety variables start with "S_". Verify that names of global safety variables start with "GS_". Verify that names of safety inputs start with "IS_". Verify that names of safety outputs start with "OS_". Verify that names of non-safety variables do not start with either "S_", "GS_", "IS_" or "OS_".	
Verify that names of global variable lists containing non-safety variables do not start with S_.	
Verify that names of global variable lists containing safety variables start with S_.	
For each non-safety POU, verify that it does not write to any safety variable.	

4.6 AC500-S libraries

4.6.1 Overview

The following safety libraries are certified by TÜV SÜD and are allowed to be used with AC500-S safety PLC.

Table 14: Safety libraries

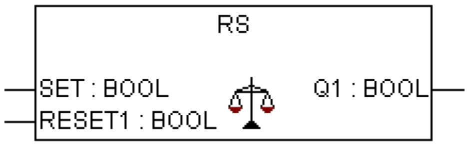
Library name / version	Library CRC	Description
Safety_Standard.lib Version 2.3, which is an exact copy of the proven-in-use CODESYS standard.lib (Version 2.3)	fd5d3581	Standard functions of CODESYS *[4]*.
Safety_SysLibTime.lib Version 2.4.0.6	672b8325	Internal time system library **(Internal use only!)**

Library name / version	Library CRC	Description
SafetyBase_PROFIsafe_LV200_ AC500_V22.lib Version 2.0.0	1d881052	PROFIsafe F-Host and Safety I/O base functions Old versions: ● SafetyBase_PROFIsafe_AC500_V22_Ext.lib, version 1.0.1, library CRC: f34d9a48. ● SafetyBase_PROFIsafe_AC500_V22.lib, version 1.0.0, library CRC: 7f64e267, license activation with PS501-S License Enabling Package. ● SafetyBase_PROFIsafe_AC500_V22.lib, version 1.0.0, library CRC: c688eb23, special OEM version of PROFIsafe library. **Note:** Old versions are NOT for use in new AC500-S customer projects.
SafetyBlocks_PLCopen_AC500_v22.lib Version 1.0.0	b6e0bc60	PLCopen Safety library
SafetyDeviceExt_LV100_ PROFIsafe_AC500_V27.lib Version 1.0.0	2eadeae9	PROFIsafe F-Device function on safety CPU For safety CPU firmware V2.0.0 or higher.
SafetyExt2_LV100_AC500_V27.lib Version 1.0.0	f3eb2fbc	Safety functions for safety CPU: ● triggering of safe stop ● reading of configured maximum power dip value ● reading of boot project CRC These are additional functions to those available in SafetyExt_AC500_V22.lib. For safety CPU firmware V2.0.0 or higher.
SafetyExt_AC500_V22.lib Version 1.0.0	72a88162	Safety functions for safety CPU cycle monitoring, under- and overvoltage supervision, data exchange with non-safety CPU, user data storage in the flash memory, etc.
SafetyUtil_CoDeSys_AC500_V22.lib Version 1.0.0	6b29c54	Internal safety utilities of the safety CPU **(Internal use only!)**
SysLibCallback.lib Version 2.4.0.6	62ad210d	Internal CODESYS library (not shown in Library Manager) **(Internal use only!)**
Target_AC500_V22.lib Version 3.4.0.6	8daa436	Internal AC500 library (not shown in Library Manager) **(Internal use only!)**

4.6.2 Safety_Standard.lib

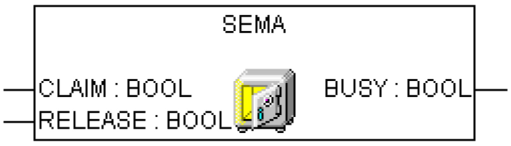
Only a short description is provided for standard CODESYS POUs from Safety_Standard.lib. For more detailed information about CODESYS standard functions refer to ⇘ *[4]*.

RS



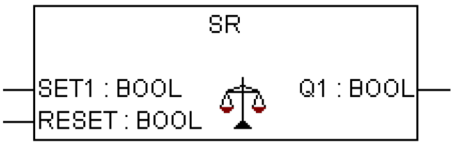
Bistable function, reset dominant
Q1 = NOT RESET1 AND (SET OR Q1)

SEMA



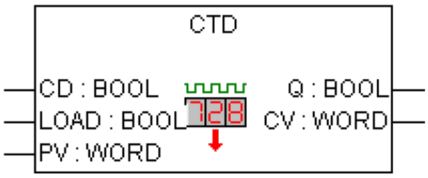
Software semaphore. Interruptible!
BUSY is TRUE, if there was a call with CLAIM = TRUE, but no call with RELEASE = TRUE.
CLAIM = TRUE sets BUSY = TRUE;
RELEASE = TRUE sets BUSY = FALSE;

SR



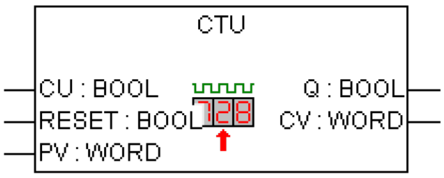
Bistable function, set dominant
Q1 = SET1 OR (NOT RESET AND Q1)

CTD



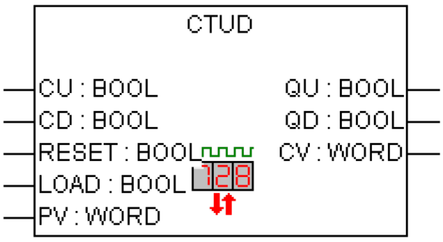
Counter down.
CV is decremented by 1 if CD has a rising edge.
Q is TRUE, if CV reached 0.

CTU



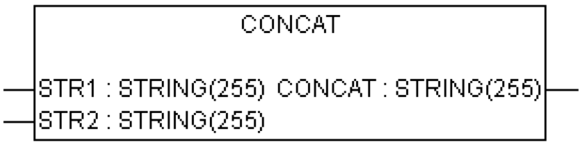
Counter up.
CV is incremented by 1 if CU has a rising edge.
Q is TRUE, if CV is reached PV.

CTUD



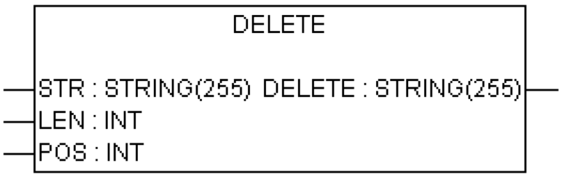
Counter up down
CV is incremented by 1 if CU has a rising edge.
CV is decremented by 1 if CD has a rising edge.
QU is TRUE, if counter is PV.
QD is TRUE, if counter is 0.

CONCAT



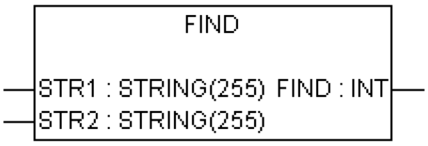
Concatenation of two strings.

DELETE



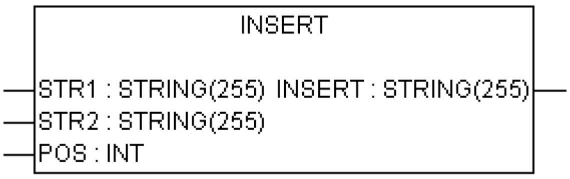
Delete LEN characters of STR, beginning at the POS-th character position.
POS = 1 is the first character.

FIND



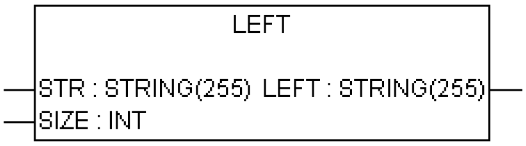
Find the character position of the beginning of the first occurrence of STR2 in STR1.
If no occurrence of STR1 is found, then the result is 0.

INSERT



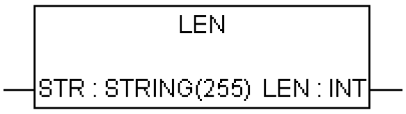
Insert STR2 into STR1 after the POS-th character position.
POS = 0 inserts before the first character.
POS = 1 inserts after the first character.

LEFT



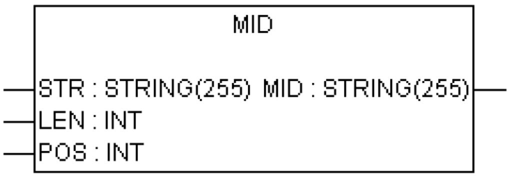
Return leftmost SIZE characters of STR.

LEN



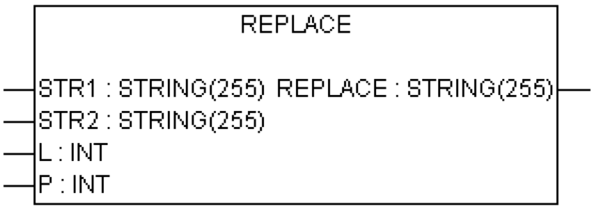
String length function.
Returns the number of characters in STR.

MID



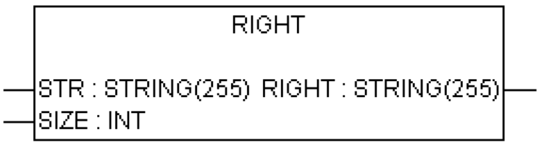
Return LEN characters of STR, beginning at the POS-th character position.
POS = 1 is the first character.

REPLACE



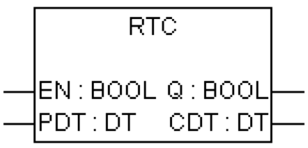
Replaces L characters of STR1 by STR2,
starting at the POS-th character position and returns the new string.
POS = 1 is the first character.

RIGHT



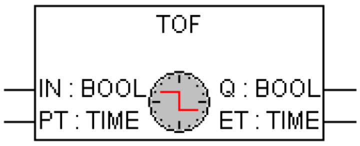
Returns rightmost SIZE characters of STR.

RTC



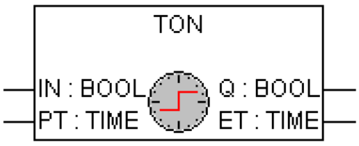
Sets CDT to PDT when rising edge in EN and starts increasing CDT.
With EN = FALSE, CDT set to DT#1970-01-01-00-00:00

TOF



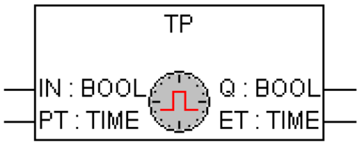
Timer of delay.
Q is FALSE, PT milliseconds after IN had a falling edge.

TON



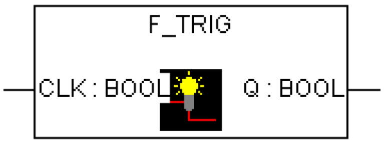
Timer on delay.
Q is TRUE, PT milliseconds after IN had a rising edge.

TP



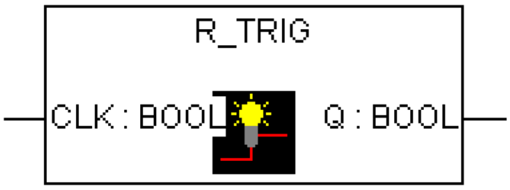
Timer pulse.
Q produces a high-signal with the length of PT on every rising edge on IN.

F_TRIG



Falling edge detection.

R_TRIG



Rising edge detection.

4.6.3 SafetyBase_PROFIsafe_LV200_AC500_V22.lib

This library includes a PROFIsafe stack implementation (PROFISAFESTACK POU), which is a main F-Host component.

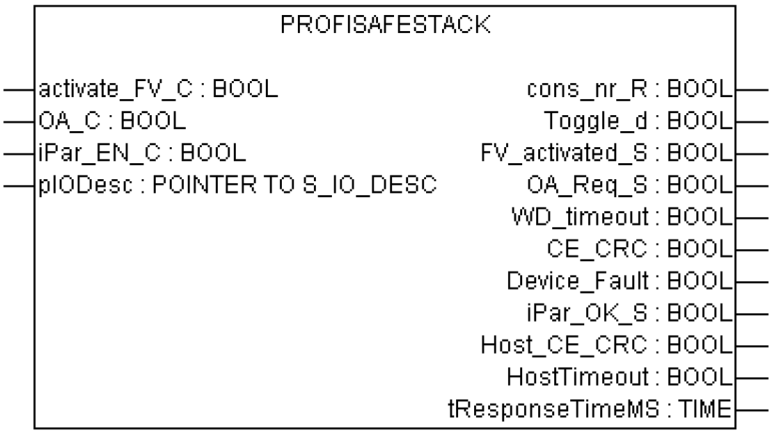
NOTICE!
Old versions

– SafetyBase_PROFIsafe_AC500_V22_Ext.lib, version 1.0.1, library CRC: f34d9a48
– SafetyBase_PROFIsafe_AC500_V22.lib, version 1.0.0, library CRC: 7f64e267, license activation with PS501-S License Enabling Package
– SafetyBase_PROFIsafe_AC500_V22.lib, version 1.0.0, library CRC: c688eb23, special OEM version of PROFIsafe library

are NOT for use in new AC500-S customer projects.

NOTICE!
Loop-back check via bit 7 in status / control byte of PROFIsafe telegram is implemented, which means that no further considerations against systematic loop-back configuration errors shall be performed by end-users (refer to *www.profisafe.net* for further details).

DANGER!
Not more than one communication error (CE_CRC or Host_CE_CRC output signals become equal to TRUE) per 100 hours is allowed to be acknowledged by the operator using OA_C input signal without consulting the responsible safety personnel (refer to *www.profisafe.net* for further details).



This function block represents PROFIsafe F-Host instance to control and monitor the status of the given F-Device (safety I/O, etc.) *[3]*.

Table 15: FB name: PROFISAFESTACK

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
activate_FV_C	BOOL	FALSE	Command (= TRUE) to activate fail-safe values in F-Device or (= FALSE) for normal F-Device operation
OA_C	BOOL	FALSE	Command (= TRUE) for operator acknowledgment and resume of safety function by F-Device
iPar_EN_C	BOOL	FALSE	This variable TRUE allows a safety control program to switch the F-Device into a mode during which it will accept iParameters. This mode is not supported by AC500-S safety I/O modules (DI581-S, DX581-S, AI581-S and SM560-S-FD-1 / SM560-S-FD-4)

Name	Data type	Initial value	Description, parameter values
pIODesc	POINTER	NULL	Internal input parameter **(internal use only!)**
VAR_OUTPUT			
cons_nr_R	BOOL	FALSE	**This parameter is for debugging purposes only.** It is set when the F-Device has reset its consecutive number counter in PROFIsafe communication ⇘ *[3]*.
Toggle_d	BOOL	FALSE	**This parameter is for debugging purposes only.** It is a device-based toggle bit indicating a trigger to increment the virtual consecutive number within the F-Host ⇘ *[3]*.
FV_activated_S	BOOL	FALSE	With input devices this variable indicates if TRUE that the driver is delivering fail-safe values "0" to the F-Host program for every input value. With output devices this variable indicates if TRUE that every output is set to fail-safe values "0" (default behavior) or F-Output device specific value controlled by the "activate_FV" signal ⇘ *[3]*.
OA_Req_S	BOOL	FALSE	This variable indicates a request for acknowledgment prior to the resumption of a safety function. In case the F-Host driver or F-Device detects a communication error or F-Device fault, fail-safe values will be activated. F-Device driver then sets the variable OA_Req_S (= TRUE) as soon as the fault/error has been eliminated and operator acknowledgment is possible. Once the acknowledgment occurred (OA_C = TRUE) the F-Device driver will reset the request variable OA_Req_S (= FALSE) ⇘ *[3]*.
WD_timeout	BOOL	FALSE	**This parameter is for debugging purposes only.** It is set to TRUE if the F-Device is recognizing a communication failure, i.e. if the watchdog time in the F-Device is exceeded ⇘ *[3]*.
CE_CRC	BOOL	FALSE	**This parameter is for debugging purposes only.** It is set if the F-Device is recognizing a communication failure, i.e. if the consecutive number is wrong (detected via CRC2 error in V2-mode) or the data integrity is violated (CRC error) ⇘ *[3]*.
Device_Fault	BOOL	FALSE	This parameter is set to TRUE if there is a malfunction in the F-Device (e.g., under- or overvoltage) ⇘ *[3]*.
iPar_OK_S	BOOL	FALSE	This parameter is set to TRUE when F-Device has new parameter values assigned ⇘ *[3]*.
Host_CE_CRC	BOOL	FALSE	**This parameter is for debugging purposes only.** This parameter is set to TRUE if communication fault (CRC error on F-Host side) occurred.

Name	Data type	Initial value	Description, parameter values
HostTimeout	BOOL	FALSE	**This parameter is for debugging purposes only.** This parameter is set to TRUE if communication fault (timeout on F-Host side) occurred.
tResponseTimeMS	TIME	16#0000	**This parameter is for debugging purposes only.** It represents the current response time for F-Device in ms. This value shall be smaller than the defined F_WD_Time parameter for the given F-Device. If not, then the passivation of the given F-Device will happen.

The FB instances for all F-Devices are automatically generated and can be found in CODESYS Safety project in *“Resources ➔ Global Variables ➔ PROFIsafe”* (Fig. 91 on page 198). These FB instances, as normal global variables, can be accessed by end-users from their safety application programs.

DANGER!

To avoid unintended behavior, e.g., unintended restart, of 3rd party PROFIsafe devices, one shall pay a special attention to the description of PROFIsafe Device_Fault bit in the safety user manual for those devices.

It is highly recommended to continuously supervise Device_Fault bit of 3rd party PROFIsafe actuator devices like valves, etc. to avoid unintended restart of those after, e.g. power failure. If Device_Fault = 1 is detected for such devices, then the safety application shall passivate the module with activate_FV_C = 1. The permission for restart (activate_FV_C = 0) shall be handled in the safety application using the functionality similar to that of FB SF_OutControl ↬ *Chapter 4.6.4.17 “SF_OutControl” on page 289*.

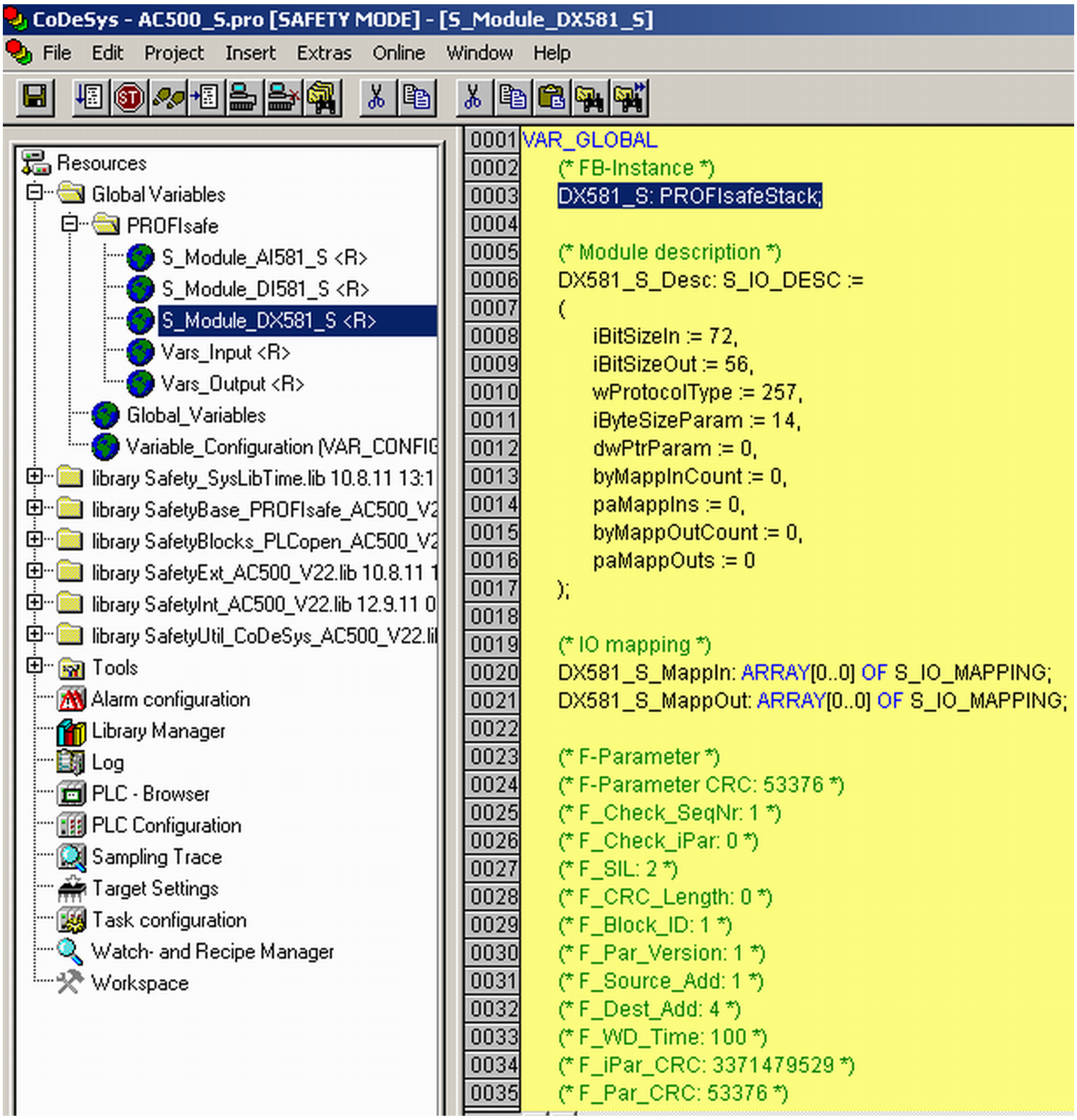


Fig. 91: FB instances for F-Devices

Note, that SafetyBase_PROFIsafe_LV200_AC500_V22.lib library also includes a number of internal POUs (GetWord, MappingIn, MappingOut and SMemCpy) related to safety I/O handling. **These POUs are for internal use only!**

4.6.4 SafetyBlocks_PLCopen_AC500_v22.lib

A list of supported PLCopen Safety POUs is presented in the following sub-chapters. The developed PLCopen Safety POUs are based on ↯ *[9]*.

NOTICE!
The referenced standards in the following sub-chapters are used for information only:

– EN 954-1:1996
– IEC 60204-1 Ed. 5.0:2003
– IEC 61496-1:2004
– IEC 62046/Ed.1:2005
– ISO 12100-2:2003
– MRL 98/37/EC, Annex I
– EN 418:1992
– EN 574:1996
– EN 1088:1995
– EN 953:1997

Use for functional safety certification the newest functional safety standards *Chapter 1.8 “Applicable standards” on page 12*.

4.6.4.1 Introduction

Generic parameters and diagnostic codes of PLCopen Safety POUs are presented below.

Table 16: General input parameters

Name	Type	Description
Activate	BOOL	Variable or constant. Activation of the FB. Initial value is FALSE. This parameter can be connected to the variable, which represents the status (active or not active) of the relevant safety device. This ensures no irrelevant diagnostic information is generated if a device is disabled. If FALSE, all output variables are set to the initial values. If no device is connected, a static TRUE signal must be assigned.
S_StartReset	BOOL	Variable or constant. FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the automatic circuit reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) start-up does not occur.

Name	Type	Description
S_AutoReset	BOOL	Variable or constant. FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the automatic circuit reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
Reset	BOOL	Variable. Initial value is FALSE. Depending on the function, this input can be used for different purposes: ● Reset of the state machine, coupled error and status messages as indicated via DiagCode, when the error cause has been removed. This reset behavior is designed as an error reset. ● Manual reset of a "restart interlock" by the operator (refer to EN 954-1). This reset behavior is designed as a functional reset. ● Additional FB-specific reset functions. This function is only active on a signal change from FALSE to TRUE. A static TRUE signal causes no further actions, but may be detected as an error in some FBs. The appropriate meaning must be described in every FB.

Table 17: General output parameters

Name	Type	Description
Ready	BOOL	If TRUE, indicates that the FB is activated and the output results are valid (same as the "POWER" LED of a safety relay). If FALSE, the FB is not active and the program is not executed. Useful in debug mode or to activate/deactivate additional FBs, as well as for further processing in the functional program.
Error	BOOL	Error flag (same as "K1/K2" LED of a safety relay). When TRUE, indicates that an error has occurred, and the FB is in an error state. The relevant error state is mirrored at the DiagCode output. If FALSE, there is no error and the FB is in another state. This again is mirrored by DiagCode (this means that DiagCode must be set in the same cycle as the state change). Useful in debug mode as well as for further processing in the functional program.
DiagCode	WORD	Diagnostic register. All states of the FB (active, not active and error) are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes. Only one consistent code is represented at the same time. In the event of multiple errors, the DiagCode output indicates the first detected error. *Table 18 “General diagnostic code ranges” on page 201* *Table 19 “System or device-specific codes” on page 201* *Table 20 “General diagnostic codes” on page 201* Useful in debug mode as well as for further processing in the functional program.

A transparent and unique diagnostic concept forms the basis of all function blocks. Thus, it is ensured, that, regardless of the supplier's implementation, uniform diagnostic information is available to the user in the form of DiagCode. If no error is present, the internal status of the function block (state machine) is indicated. An error is indicated via a binary output (error). Detailed information about internal or external function block errors can be obtained via DiagCode. The function block must be reset via the different reset inputs.

Suppliers may add additional interfaces via function blocks with supplier-specific diagnostic information.

Table 18: General diagnostic code ranges

DiagCode	Description
$0000\_0000\_0000\_0000_{bin}$	The FB is not activated or safety CPU is halted.
$10xx\_xxxx\_xxxx\_xxxx_{bin}$	Shows that the activated FB is in an operational state without an error. X = FB-specific code.
$11xx\_xxxx\_xxxx\_xxxx_{bin}$	Shows that the activated FB is in an error state. X = FB-specific code.

Table 19: System or device-specific codes

DiagCode	Description
$0xxx\_xxxx\_xxxx\_xxxx_{bin}$	X = system or device-specific message. This information contains the diagnostic information for the system or device, and is mapped directly to the DiagCode output. (Note: 0000_{hex} is reserved)

Table 20: General diagnostic codes

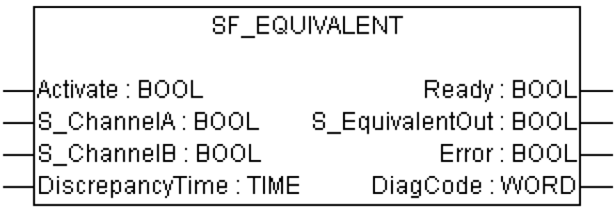
DiagCode	Description
$0000\_0000\_0000\_0000_{bin}$ 0000_{hex}	The FB is not activated. This code represents the Idle state. For a generic example, the I/O setting could be: Activate = FALSE S_In = FALSE or TRUE Ready = FALSE Error = FALSE S_Out = FALSE
$0111\_1111\_1111\_1111_{bin}$ $7FFF_{hex}$	Value 16#7FFF at DiagCode output of PLCopen Safety function blocks indicates an internal error. Contact ABB technical support. **Note:** This is a manufacturer-specific value defined by AC500-S safety PLC.
$1000\_0000\_0000\_0000_{bin}$ 8000_{hex}	The FB is activated without an error or any other condition that sets the safety output to FALSE. This is the default operational state where the S_Out safety output = TRUE in normal operation. For a generic example, the I/O setting could be: Activate = TRUE S_In = TRUE Ready = TRUE Error = FALSE S_Out = TRUE

DiagCode	Description
1000_0000_0000_0001$_{bin}$ 8001$_{hex}$	An activation has been detected by the FB and the FB is now activated, but the S_Out safety output is set to FALSE. This code represents the Init state of the operational mode. For a generic example, the I/O setting could be: Activate = TRUE S_In = FALSE or TRUE Ready = TRUE Error = FALSE S_Out = FALSE
1000_0000_0000_0010$_{bin}$ 8002$_{hex}$	The activated FB detects a safety demand, e.g., S_In = FALSE. The safety output is disabled. This is an operational state where the S_Out safety output = FALSE. For a generic example, the I/O setting could be: Activate = TRUE S_In = FALSE Ready = TRUE Error = FALSE S_Out = FALSE
1000_0000_0000_0011$_{bin}$ 8003$_{hex}$	The safety output of the activated FB has been disabled by a safety demand. The safety demand is now withdrawn, but the safety output remains FALSE until a reset condition is detected. This is an operational state where the S_Out safety output = FALSE. For a generic example, the I/O setting could be: Activate = TRUE S_In = FALSE => TRUE (continuing with static TRUE) Ready = TRUE Error = FALSE S_Out = FALSE

Note: If there are more operational states where safety output = TRUE, the next available DiagCode number will be assigned for subsequent states.

4.6.4.2 SF_Equivalent

Standards	Requirements
EN 954-1:1996	6.2 General safety principles, Idle current 6.2 Error detection for category 3 and 4

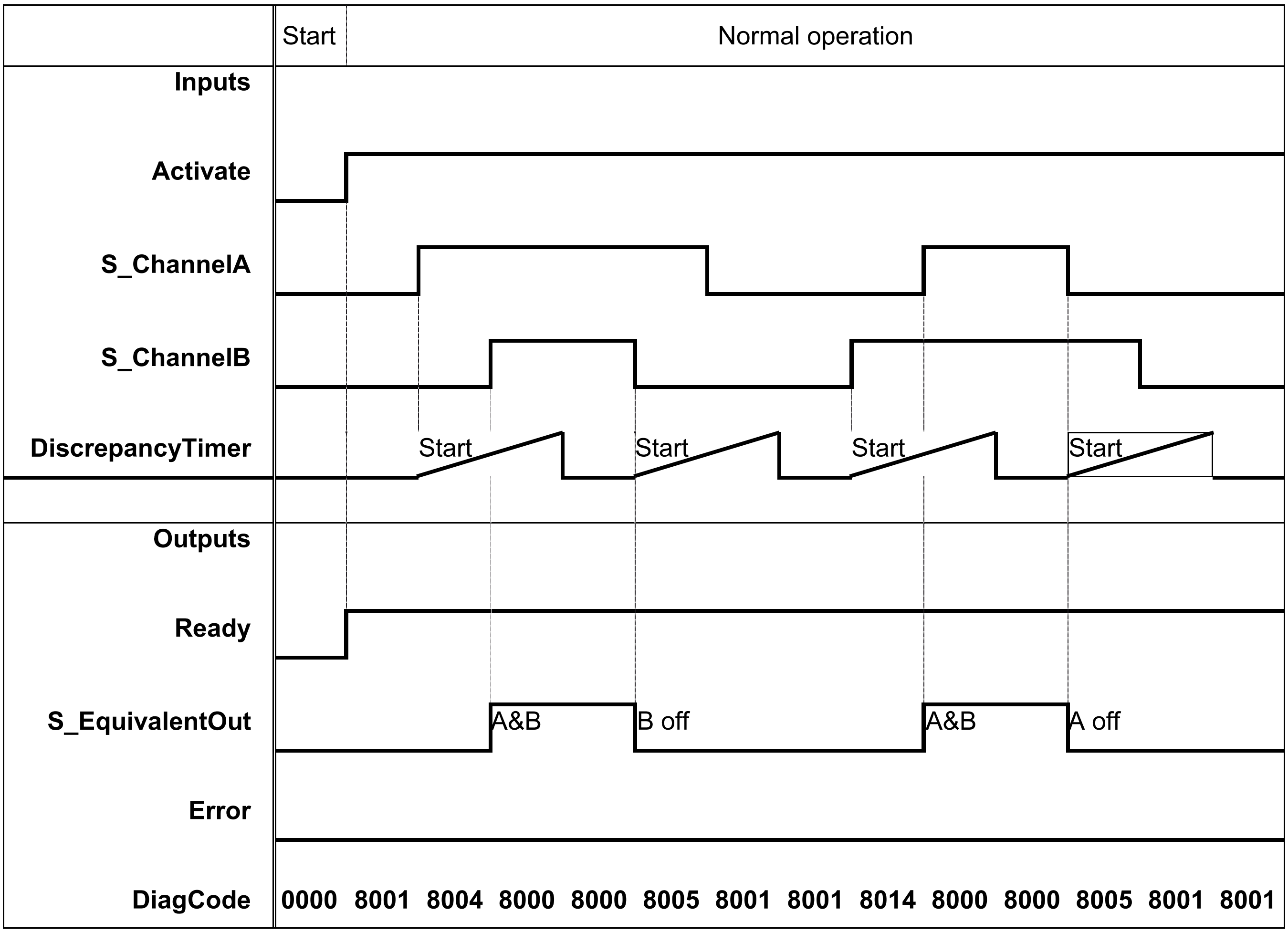


This function block converts two equivalent BOOL inputs (both NO or NC) to one BOOL output, including discrepancy time monitoring. This FB should not be used stand-alone since it has no restart interlock. It is required to connect the output to other safety FBs.

Table 21: FB name: SF_Equivalent

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_ChannelA	BOOL	FALSE	Variable. Input A for logical connection. FALSE: Contact A open TRUE: Contact A closed.
S_ChannelB	BOOL	FALSE	Variable. Input B for logical connection. FALSE: Contact B open TRUE: Contact B closed.
DiscrepancyTime	TIME	T#0ms	Constant. Maximum monitoring time for discrepancy status of both inputs.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_EquivalentOut	BOOL	FALSE	Safety related output FALSE: Minimum of one input signal = "FALSE" or status change outside of monitoring time. TRUE: Both input signals "active" and status change within monitoring time.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Typical timing diagrams



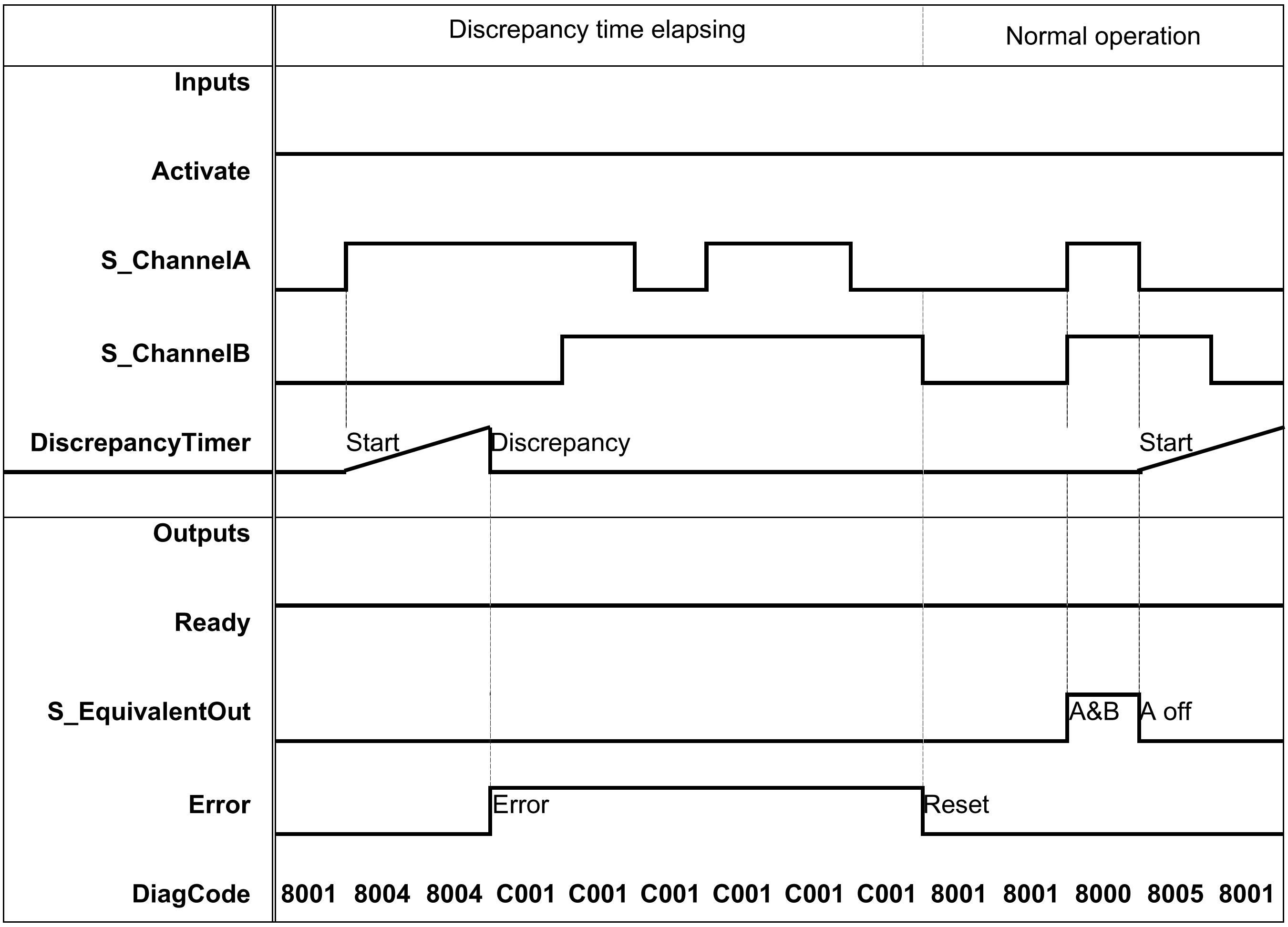


Fig. 92: Typical timing diagram for SF_Equivalent

The function block monitors the discrepancy time between channel A and B, when switching to TRUE and also when switching to FALSE.

Error behavior S_EquivalentOut is set to FALSE. Error is set to TRUE. DiagCode indicates the error states. There is no reset defined as an input is coupled with the reset of an error. If an error occurs in the inputs, a new set of inputs with correct S_EquivalentOut must be able to reset the error flag. (Example: if a switch is faulty and replaced, using the switch again results in a correct output.)

Function block-specific error and status codes

Table 22: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Error 1	Discrepancy time elapsed in state 8004. Ready = TRUE S_EquivalentOut = FALSE Error = TRUE
C002	Error 2	Discrepancy time elapsed in state 8014. Ready = TRUE S_EquivalentOut = FALSE Error = TRUE
C003	Error 3	Discrepancy time elapsed in state 8005. Ready = TRUE S_EquivalentOut = FALSE Error = TRUE

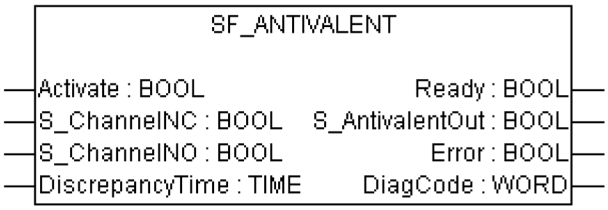
Table 23: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EquivalentOut = FALSE Error = FALSE
8001	Init	An activation has been detected by the FB and the FB is now activated. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE
8000	Safety Output Enabled	The inputs switched to TRUE in equivalent mode. Ready = TRUE S_EquivalentOut = TRUE Error = FALSE
8004	Wait for Channel B	Channel A has been switched to TRUE - waiting for channel B; discrepancy timer started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8014	Wait for Channel A	Channel B has been switched to TRUE - waiting for channel A; discrepancy timer started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE
8005	From Active Wait	One channel has been switched to FALSE; waiting for the second channel to be switched to FALSE, discrepancy timer started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE

4.6.4.3 SF_Antivalent

Standards	Requirements
EN 954-1:1996	6.2 General safety principles, Idle current 6.2 Error detection for category 3 und 4



This function block converts two antivalent BOOL inputs (NO/NC pair) to one BOOL output with discrepancy time monitoring. This FB should not be used stand-alone since it has no restart interlock. It is required to connect the output to other safety FBs.

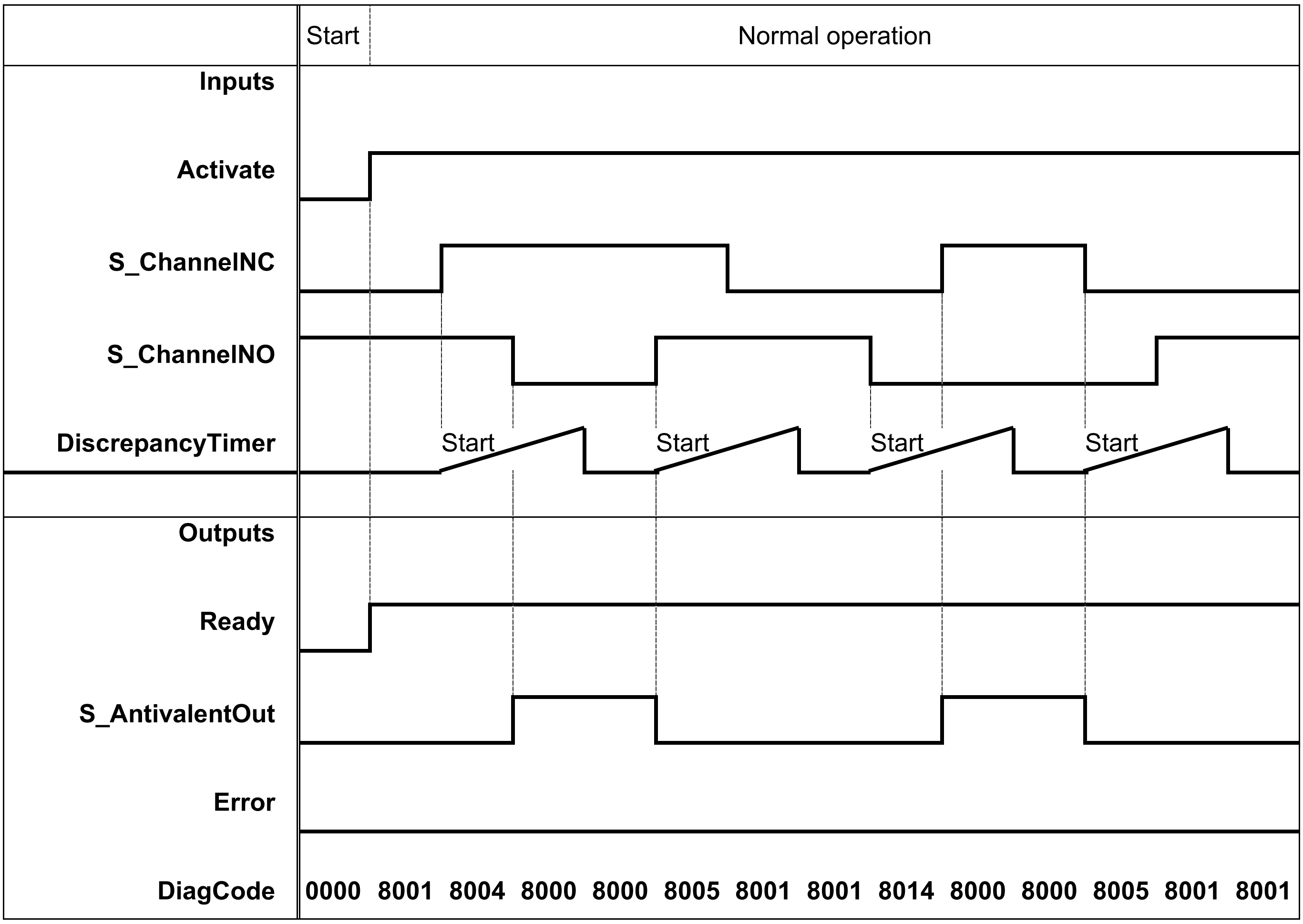
Table 24: FB name: SF_Antivalent

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_ChannelNC	BOOL	FALSE	Variable. NC stands for normally closed. Input for NC connection. FALSE: NC contact open. TRUE: NC contact closed.
S_ChannelNO	BOOL	TRUE	Variable. NO stands for normally open. Input for NO connection. FALSE: NO contact open TRUE: NO contact closed
DiscrepancyTime	TIME	T#0ms	Constant. Maximum monitoring time for discrepancy status of both inputs.

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_AntivalentOut	BOOL	FALSE	Safety related output FALSE: Minimum of one input signal "not active" or status change outside of monitoring time. TRUE: Both inputs signals "active" and status change within monitoring time.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Notes: "Antivalent" means that during normal operation, the two inputs are in opposite states at the same time. This is sometimes called "complementary" or "non-equivalent".

Typical timing diagrams



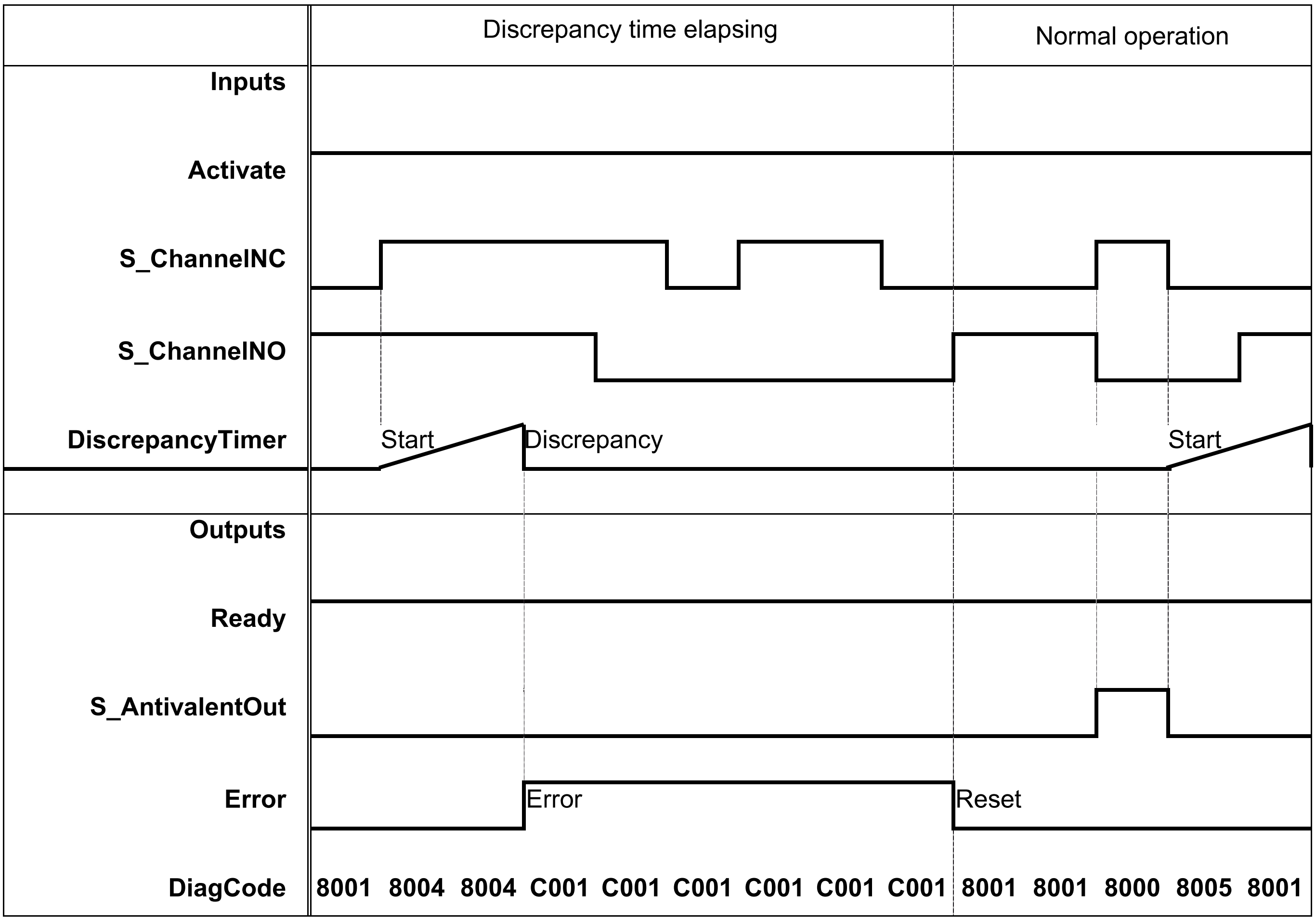


Fig. 93: Typical timing diagram for SF_Antivalent

The function block monitors the discrepancy time between channel NO and channel NC.

Error behavior

The output S_AntivalentOut is set to FALSE. Error is set to TRUE. DiagCode indicates the error states.

There is no reset defined as an input is coupled with the reset of an error. If an error occurs in the inputs, one new set of inputs with the correct value must be able to reset the error flag. (Example: if a switch is faulty and replaced, using the switch again results in a correct output.)

Function block-specific error and status codes

Table 25: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Error 1	Discrepancy time elapsed in state 8004. Ready = TRUE S_AntivalentOut = FALSE Error = TRUE
C002	Error 2	Discrepancy time elapsed in state 8014. Ready = TRUE S_AntivalentOut = FALSE Error = TRUE
C003	Error 3	Discrepancy time elapsed in state 8005. Ready = TRUE S_AntivalentOut = FALSE Error = TRUE

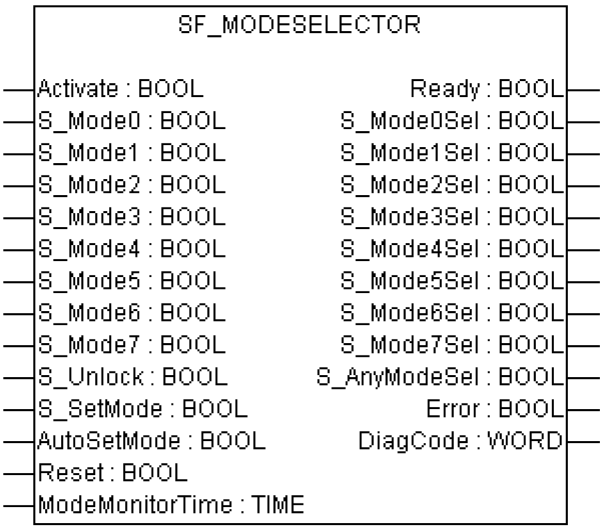
Table 26: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AntivalentOut = FALSE Error = FALSE
8001	Init	An activation has been detected by the FB and the FB is now activated. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8000	Safety Output Enabled	The inputs switched to the Active state in antivalent mode. Ready = TRUE S_AntivalentOut = TRUE Error = FALSE
8004	Wait for NO	ChannelNC has been switched to TRUE - waiting for ChannelNO to be switched to FALSE; discrepancy timer started. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8014	Wait for NC	ChannelNO has been switched to FALSE - waiting for ChannelNC to be switched to TRUE; discrepancy timer started. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8005	From Active Wait	One channel has been switched to inactive; waiting for the second channel to be switched to inactive too. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE

4.6.4.4 SF_ModeSelector

Standards	Requirements
MRL 98/37/EC, Annex I	1.2.3. Starting ... It must be possible to start machinery only by voluntary actuation of a control provided for the purpose ...The same requirement applies: ... - when effecting a significant change in the operating conditions ... 1.2.5 ... mode selector which can be locked in each position. Each position of the selector must correspond to a single operating or control mode ...
EN ISO 12100-2:2003	4.11.10 Selection of Control and Operating Modes ... shall be fitted with a mode selector which can be locked in each position. Each position of the selector shall be clearly identifiable and shall exclusively enable one control or operating mode to be selected...
IEC 60204-1, Ed. 5.0:2003	9.2.3 Operating Modes ... When a hazardous condition can result from a mode selection, unauthorized and/or inadvertent selection shall be prevented by suitable means (e.g. key operated switch, access code). Mode selection by itself shall not initiate machine operation. A separate action by the operator shall be required. ... Indication of the selected operating mode shall be provided ...
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



This function block selects the system operation mode, such as manual, automatic, semi-automatic, etc.

Table 27: FB name: SF_ModeSelector

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
S_Mode0	BOOL	FALSE	Variable or constant. Input 0 from mode selector switch FALSE: Mode 0 is not requested by operator. TRUE: Mode 0 is requested by operator.
S_Mode1	BOOL	FALSE	Variable or constant. Input 1 from mode selector switch FALSE: Mode 1 is not requested by operator. TRUE: Mode 1 is requested by operator.
S_Mode2	BOOL	FALSE	Variable or constant. Input 2 from mode selector switch FALSE: Mode 2 is not requested by operator. TRUE: Mode 2 is requested by operator.
S_Mode3	BOOL	FALSE	Variable or constant. Input 3 from mode selector switch FALSE: Mode 3 is not requested by operator. TRUE: Mode 3 is requested by operator.
S_Mode4	BOOL	FALSE	Variable or constant. Input 4 from mode selector switch FALSE: Mode 4 is not requested by operator. TRUE: Mode 4 is requested by operator.

Name	Data type	Initial value	Description, parameter values
S_Mode5	BOOL	FALSE	Variable or constant. Input 5 from mode selector switch FALSE: Mode 5 is not requested by operator. TRUE: Mode 5 is requested by operator.
S_Mode6	BOOL	FALSE	Variable or constant. Input 6 from mode selector switch FALSE: Mode 6 is not requested by operator. TRUE: Mode 6 is requested by operator.
S_Mode7	BOOL	FALSE	Variable or constant. Input 7 from mode selector switch FALSE: Mode 7 is not requested by operator. TRUE: Mode 7 is requested by operator.
S_Unlock	BOOL	FALSE	Variable or constant. Locks the selected mode FALSE: The actual S_ModeXSel output is locked therefore a change of any S_ModeX input does not lead to a change in the S_ModeXSel output even in the event of a rising edge of SetMode. TRUE: The selected S_ModeXSel is not locked; a mode selection change is possible.
S_SetMode	BOOL	FALSE	Variable (or constant FALSE, if AutoSetMode = TRUE) Sets the selected mode Operator acknowledges the setting of a mode. Any change to new S_ModeX = TRUE leads to S_Any-ModeSel/S_ModeXSel = FALSE, only a rising Set-Mode trigger then leads to new S_ModeXSel = TRUE.
AutoSetMode	BOOL	FALSE	Constant. Parameterizes the acknowledgment mode FALSE: A change in mode must be acknowledged by the operator via SetMode. TRUE: A valid change of the S_ModeX input to another S_ModeX automatically leads to a change in S_ModeXSel without operator acknowledgment via SetMode (as long as this is not locked by S_Unlock).
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
ModeMonitorTime	TIME	T#0	Constant. Maximum permissible time for changing the selection input.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_Mode0Sel	BOOL	FALSE	Indicates that mode 0 is selected and acknowledged. FALSE: Mode 0 is not selected or not active. TRUE: Mode 0 is selected and active.

Name	Data type	Initial value	Description, parameter values
S_Mode1Sel	BOOL	FALSE	Indicates that mode 1 is selected and acknowledged. FALSE: Mode 1 is not selected or not active. TRUE: Mode 1 is selected and active.
S_Mode2Sel	BOOL	FALSE	Indicates that mode 2 is selected and acknowledged. FALSE: Mode 2 is not selected or not active. TRUE: Mode 2 is selected and active.
S_Mode3Sel	BOOL	FALSE	Indicates that mode 3 is selected and acknowledged. FALSE: Mode 3 is not selected or not active. TRUE: Mode 3 is selected and active.
S_Mode4Sel	BOOL	FALSE	Indicates that mode 4 is selected and acknowledged. FALSE: Mode 4 is not selected or not active. TRUE: Mode 4 is selected and active.
S_Mode5Sel	BOOL	FALSE	Indicates that mode 5 is selected and acknowledged. FALSE: Mode 5 is not selected or not active. TRUE: Mode 5 is selected and active.
S_Mode6Sel	BOOL	FALSE	Indicates that mode 6 is selected and acknowledged. FALSE: Mode 6 is not selected or not active. TRUE: Mode 6 is selected and active.
S_Mode7Sel	BOOL	FALSE	Indicates that mode 7 is selected and acknowledged. FALSE: Mode 7 is not selected or not active. TRUE: Mode 7 is selected and active.
S_AnyModeSel	BOOL	FALSE	Indicates that any of the 8 modes is selected and acknowledged. FALSE: No S_ModeX is selected. TRUE: One of the 8 S_ModeX is selected and active.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Note: The X in parameter names "S_ModeX" or "S_ModeXSel" is a placeholder for digits 0 to 7.

Typical timing diagrams

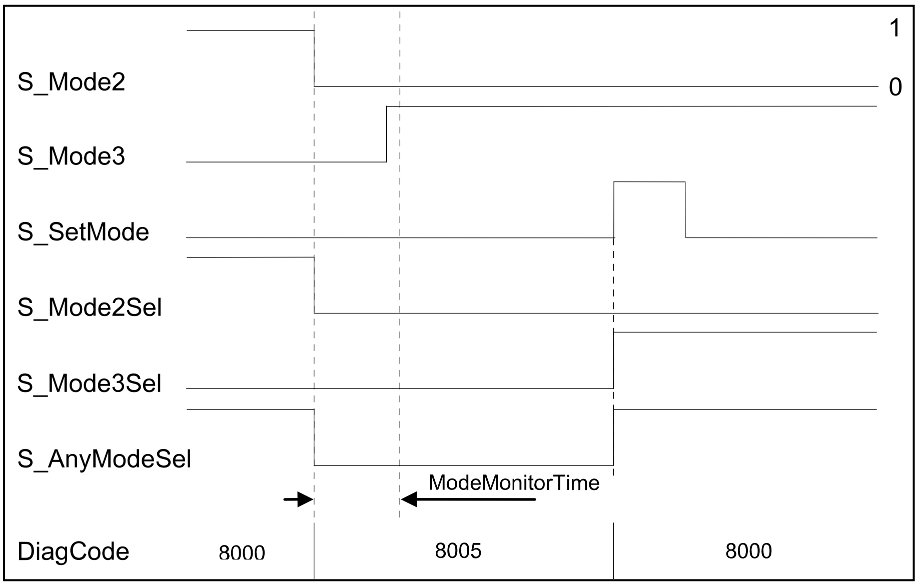


Fig. 94: Timing diagram for SF_ModeSelector, valid change in mode input with acknowledgment

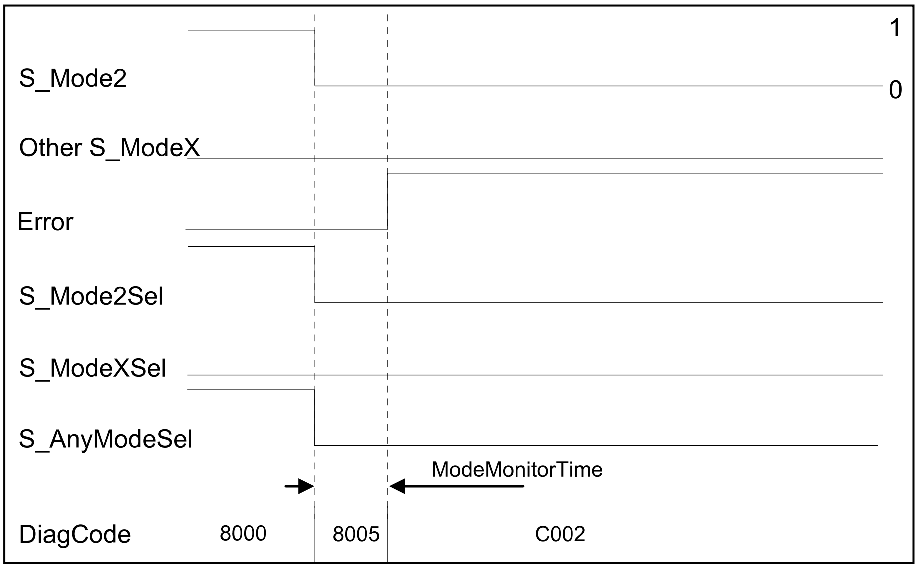


Fig. 95: Timing diagram for SF_ModeSelector, error condition 2 at mode inputs

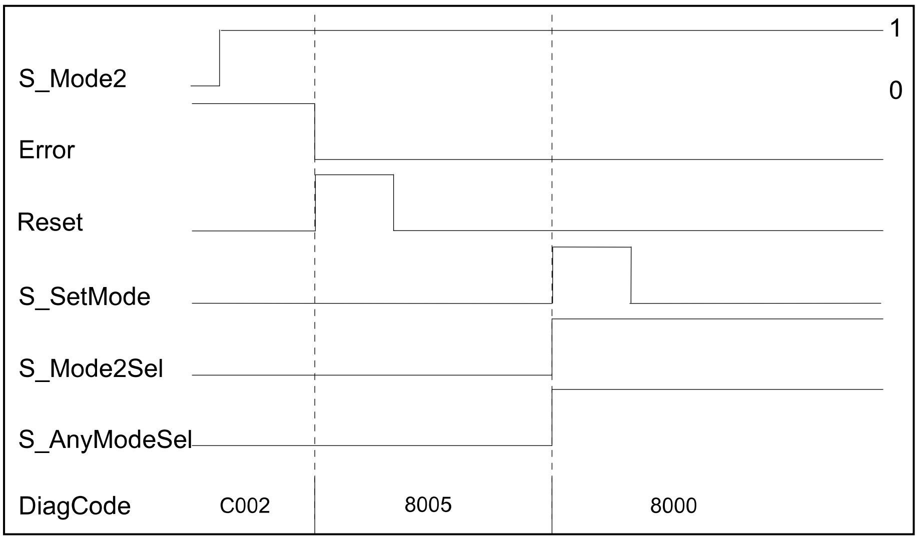


Fig. 96: Timing diagram for SF_ModeSelector, reset of error condition

The FB detects whether none of the mode inputs is selected. This invalid condition is detected after ModeMonitorTime has elapsed:

- Which restarts with each falling trigger of an S_ModeX switched mode input.
- Which is then in the ModeChanged state following activation of the FB.

In contrast, the FB directly detects whether more than one S_ModeX mode input is selected at the same time.

A static reset condition is detected when the FB is either in error state C001 or C002.

Error behavior

In the event of an error, the S_ModeXSel and S_AnyModeSel outputs are set to safe state = FALSE. The DiagCode output indicates the relevant error code and the error output is set to TRUE.

An error must be acknowledged with the rising trigger of the Reset BOOL input. The FB changes from an error state to the ModeChanged state.

Function block-specific error and status codes

Table 28: FB-specific error codes

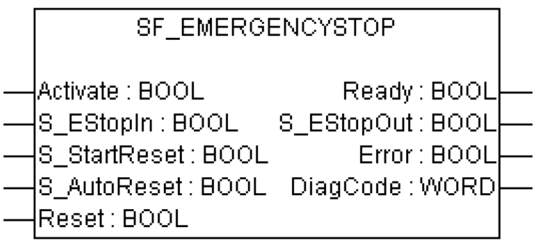
DiagCode	State name	State description and output setting
C001	Error Short-circuit	The FB detected that two or more S_ModeX are TRUE, e.g., short-circuit of cables. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C002	Error Open-circuit	The FB detected that all S_ModeX are FALSE: The period following a falling S_ModeX trigger exceeds ModeMonitorTime, e.g., open-circuit of cables. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C003	Reset Error 1	Static reset signal detected in state C001. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C004	Reset Error 2	Static reset signal detected in state C002. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE

Table 29: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8005	Mode-Changed	State after activation or when S_ModeX has changed (unless locked) or after reset of an error state. Ready = TRUE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8000	ModeSe-lected	Valid mode selection, but not yet locked. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = Selected X is TRUE, others are FALSE.
8004	ModeLocked	Valid mode selection is locked. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = Selected X is TRUE, others are FALSE.

4.6.4.5 SF_EmergencyStop

Standards	Requirements
EN 418:1992	3. Definitions 4.1.12 ... Resetting the control device shall not by itself cause a restart command.
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4 Restart following power failure/spontaneous restart
IEC 60204-1, Ed. 5.0:2003	9.2.2. Stop Functions



This function block is a safety-related function block for monitoring an emergency stop button. This FB can be used for emergency stop switch off functionality (stop category 0), or - with additional peripheral support - as emergency stop (stop category 1 or 2).

Table 30: FB name: SF_EmergencyStop

Name	**Data type**	**Initial value**	**Description, parameter values**
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
S_EStopIn	BOOL	FALSE	Safety demand input. Variable. FALSE: Demand for safety-related response (e.g., emergency stop button is engaged). TRUE: No demand for safety-related response (e.g., emergency stop button not engaged).
S_StartReset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
S_AutoReset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
Reset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
S_EStopOut	BOOL	FALSE	Output for the safety-related response. FALSE: Safety output disabled. Demand for safety-related response (e.g., emergency stop button engaged, reset required or internal errors active) TRUE: Safety output enabled. No demand for safety-related response (e.g., emergency stop button not engaged, no internal errors active).
Error	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
DiagCode	WORD	16#0000	*Table 17 "General output parameters" on page 200*

Note: The following requirements as defined in EN 418:1992 have to be fulfilled by the user:

- 4.1.4 After activation of the actuator, the emergency stop equipment shall operate in such a way that the hazard is averted or reduced automatically in the best possible manner.
- 4.1.7 The emergency stop command shall override all other commands.
- 4.1.12 Resetting the control device shall only be possible as the result of a manual action on the control device itself ... It shall not be possible to restart the machine until all control devices which have been actuated are reset manually, individually and intentionally.

Typical timing diagrams

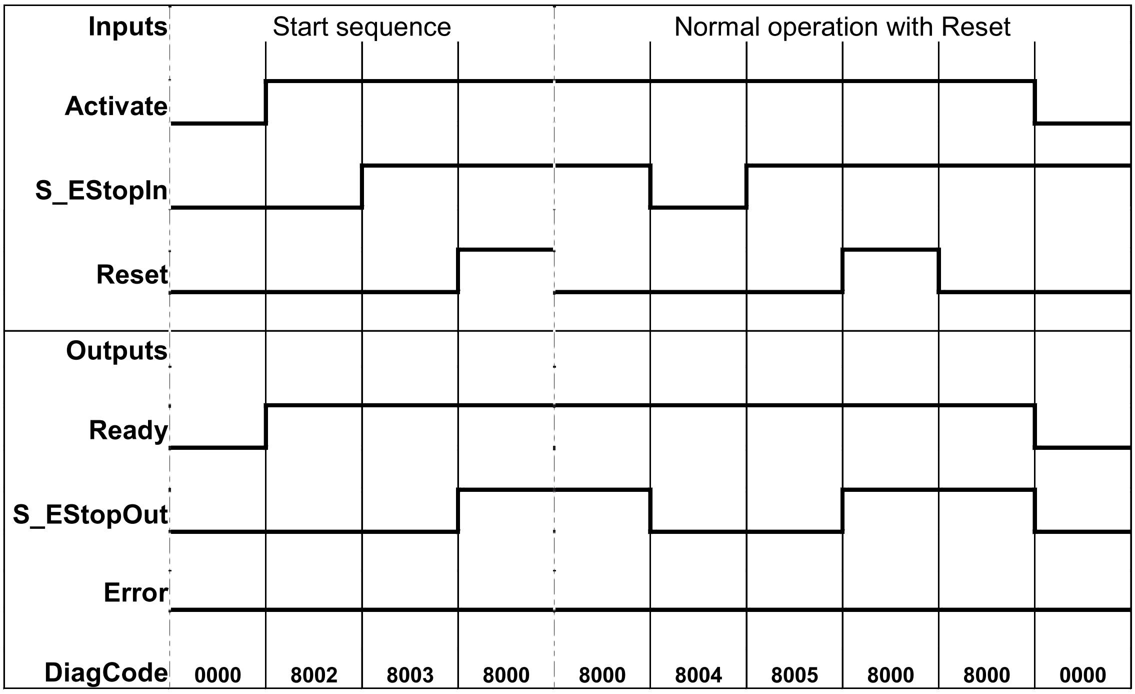


Fig. 97: Timing diagram for SF_EmergencyStop: S_StartReset = FALSE; S_AutoReset = FALSE; start, reset, normal operation, safety demand, restart

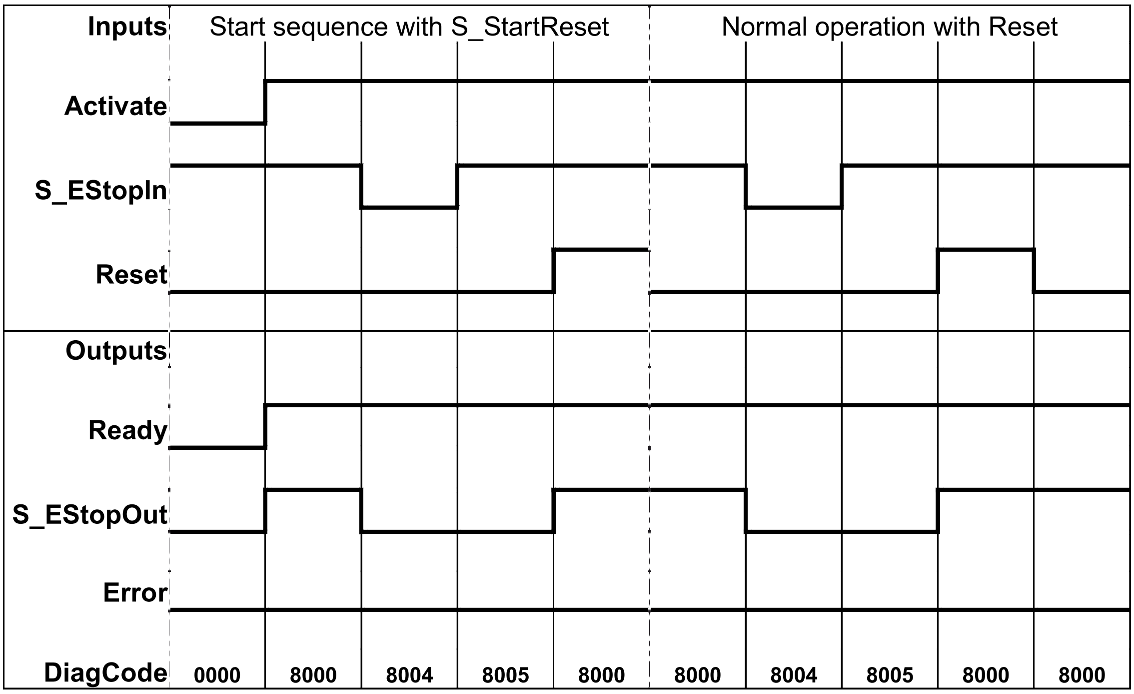


Fig. 98: Timing diagram for SF_EmergencyStop: S_StartReset = TRUE, S_AutoReset = FALSE; start, normal operation, safety demand, restart

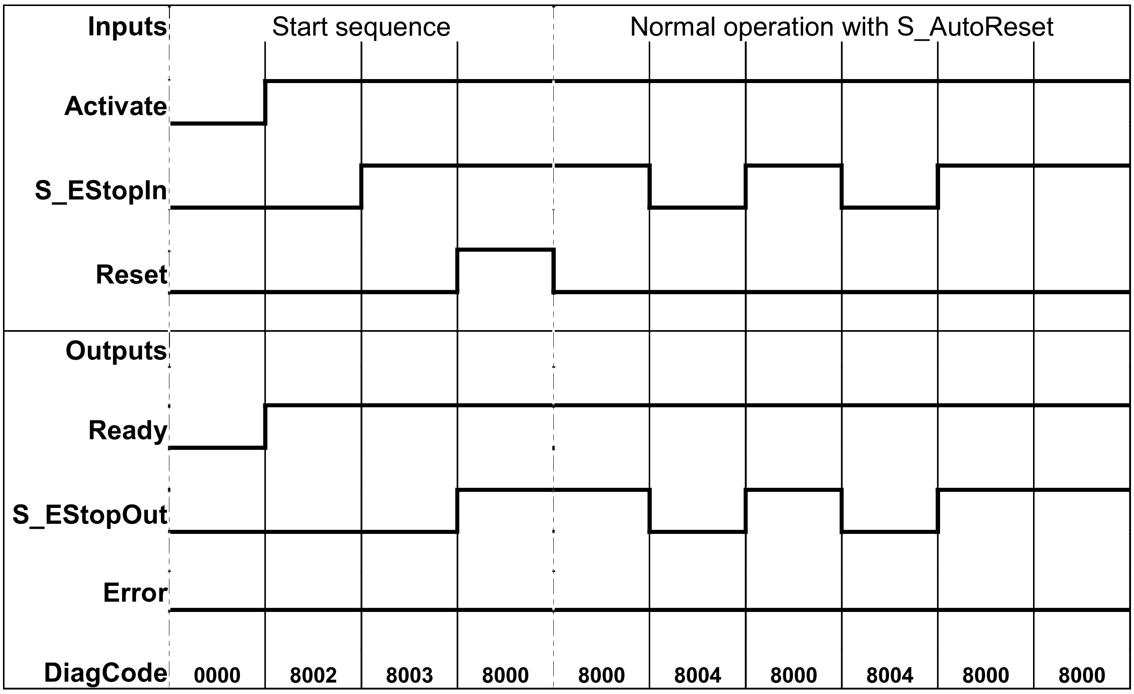


Fig. 99: Timing diagram for SF_EmergencyStop: S_StartReset = FALSE, S_AutoReset = TRUE, start, normal operation, safety demand, restart

The function block detects a static TRUE signal at Reset input.

Error behavior

S_EStopOut is set to FALSE. In case of a static TRUE signal at the Reset input, the DiagCode output indicates the relevant error code and the Error output is set to TRUE.

To leave the error states, the Reset must be set to FALSE.

Function block-specific error and status codes

Table 31: FB-specific error codes

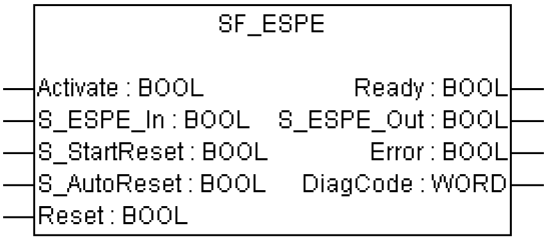
DiagCode	State name	State description and output setting
C001	Reset Error 1	Reset is TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = TRUE
C002	Reset Error 2	Reset is TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = TRUE

Table 32: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EStopOut = FALSE Error = FALSE
8001	Init	Activation is TRUE. The function block was enabled. Check if S_StartReset is required. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8002	Wait for S_EstopIn 1	Activation is TRUE. Check if Reset is FALSE and wait for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8003	Wait for Reset 1	Activation is TRUE. S_EStopIn = TRUE. Wait for rising trigger of Reset. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8004	Wait for S_EstopIn 2	Activation is TRUE. Safety demand detected. Check if Reset is FALSE and wait for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8005	Wait for Reset 2	Activation is TRUE. S_EStopIn = TRUE. Check for S_AutoReset or wait for rising trigger of Reset. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8000	Safety Output Ena-bled	Activation is TRUE. S_EStopIn = TRUE. Functional mode with S_EStopOut = TRUE. Ready = TRUE S_EStopOut = TRUE Error = FALSE

4.6.4.6 SF_ESPE

Standards	Requirements
EN IEC 61496-1:2004	A.5.1 Start Interlock: The start interlock shall prevent the OSSD(s) going to the ON-state when the electrical supply is switched on, or is interrupted and restored. A.5.2: A failure of the start interlock which causes it to go to, or remain in a permanent ON-state shall cause the ESPE to go to, or to remain in the lock-out condition. A.6.1 Restart interlock: ... The interlock condition shall continue until the restart interlock is manually reset. However, it shall not be possible to reset the restart interlock whilst the sensing device is actuated.
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



This function block is a safety-related function block for monitoring electro-sensitive protective equipment (ESPE). The function is identical to SF_EmergencyStop. The S_ESPE_Out output signal is set to FALSE as soon as the S_ESPE_In input is set to FALSE. The S_ESPE_Out output signal is set to TRUE only if the S_ESPE_In input is set to TRUE and a reset occurs. The enable reset depends on the defined S_StartReset, S_AutoReset, and Reset inputs.

If S_AutoReset = TRUE, acknowledgment is automatic.

If S_AutoReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

If S_StartReset = TRUE, acknowledgment is automatic if the PES is started for the first time.

If S_StartReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured, that no hazardous situation can occur when the PES is started.

The ESPE must be selected in respect of the product standards EN IEC 61496-1, -2 and -3 and the required categories according EN 954-1.

Table 33: FB name: SF_ESPE

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*

Name	Data type	Initial value	Description, parameter values
S_ESPE_In	BOOL	FALSE	Safety demand input. Variable. FALSE: ESPE actuated, demand for safety-related response. TRUE: ESPE not actuated, no demand for safety-related response. Safety control system must be able to detect a very short interruption of the sensor (which is specified in 61496-1: minimum 80 ms), when the ESPE is used in applications as a trip device
S_StartReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_AutoReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_ESPE_Out	BOOL	FALSE	Output for the safety-related response. FALSE: Safety output disabled. Demand for safety-related response (e.g., reset required or internal errors active). TRUE: Safety output enabled. No demand for safety-related response.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Typical timing diagrams

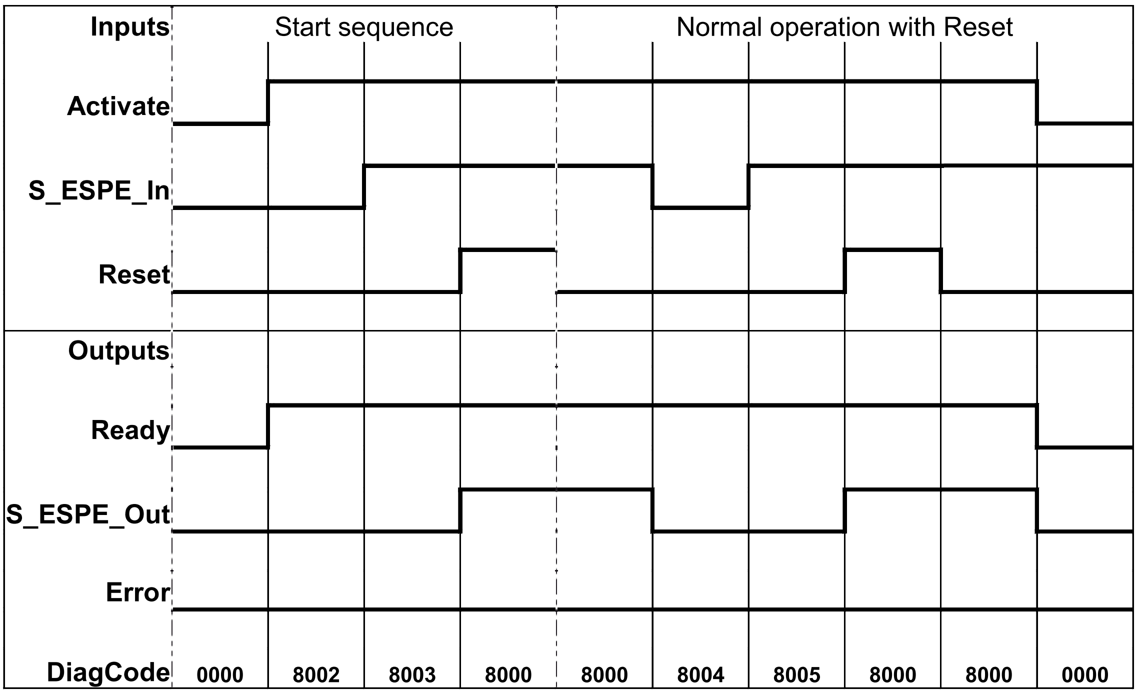


Fig. 100: Timing diagram for SF_ESPE: S_StartReset = FALSE; S_AutoReset = FALSE; start, reset, normal operation, safety demand, restart

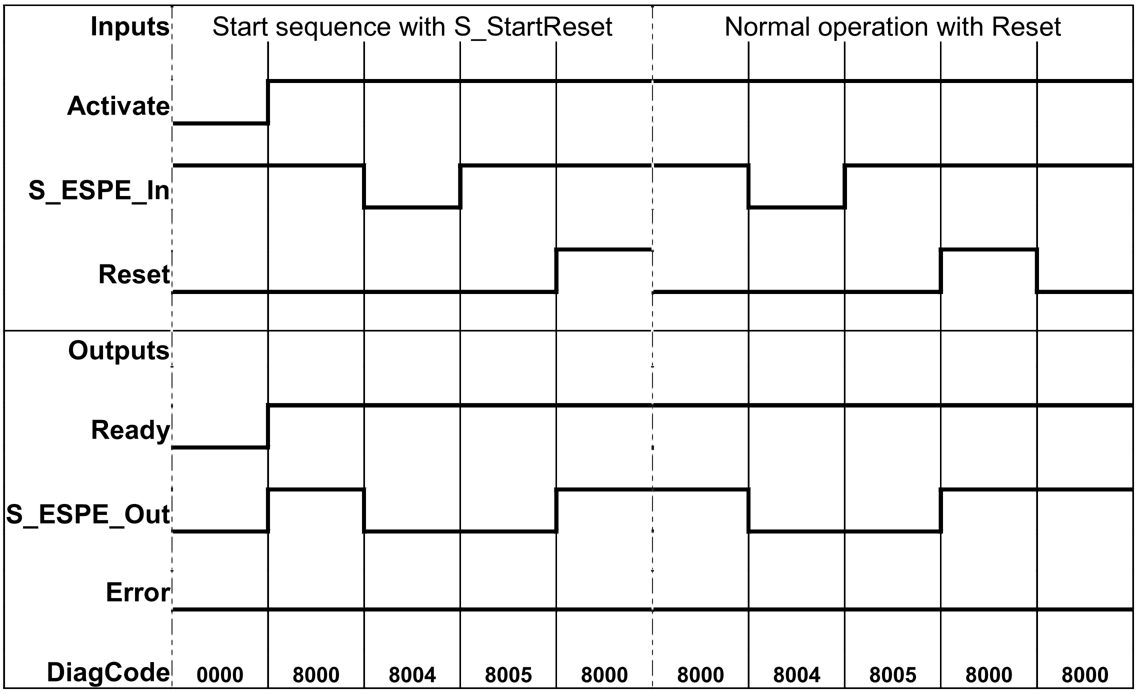


Fig. 101: Timing diagram for SF_ESPE: S_StartReset = TRUE, S_AutoReset = FALSE; start, normal operation, safety demand, restart

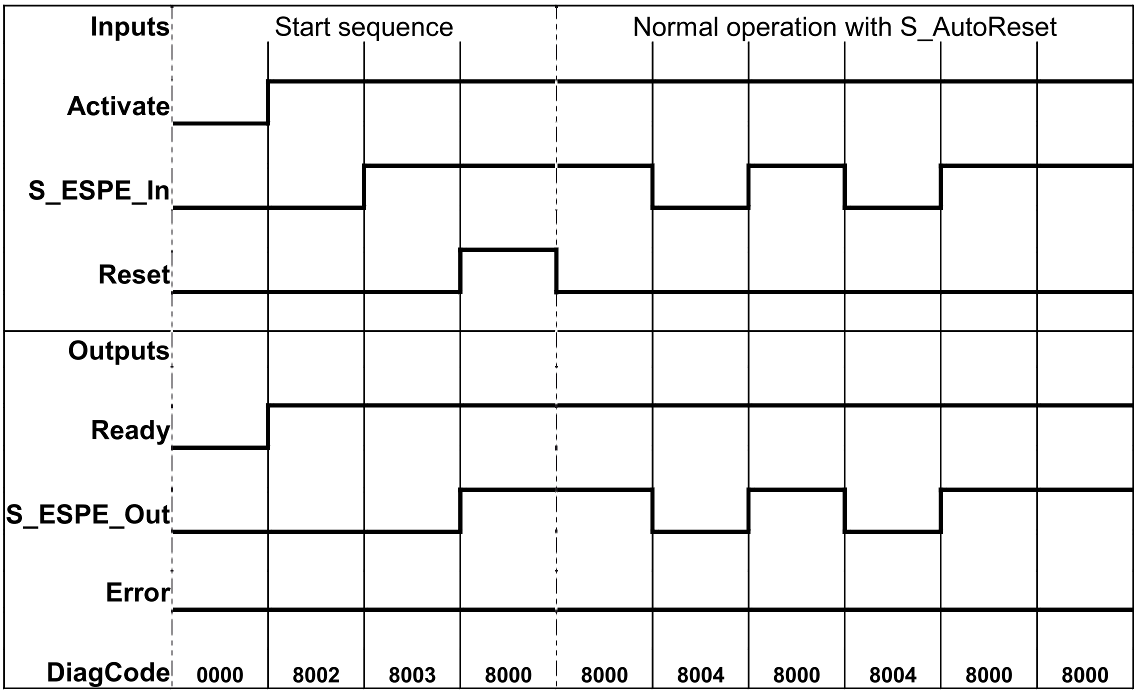


Fig. 102: Timing diagram for SF_ESPE: S_StartReset = FALSE, S_AutoReset = TRUE, start, normal operation, safety demand, restart

The function block detects a static TRUE signal at Reset input.

Error behavior

S_ESPE_Out is set to FALSE. In case of a static TRUE signal at the Reset input, the DiagCode output indicates the relevant error code and the Error output is set to TRUE.

To leave the error states, the Reset must be set to FALSE.

Function block-specific error and status codes

Table 34: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error 1	Reset is TRUE while waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = TRUE
C002	Reset Error 2	Reset is TRUE while waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = TRUE

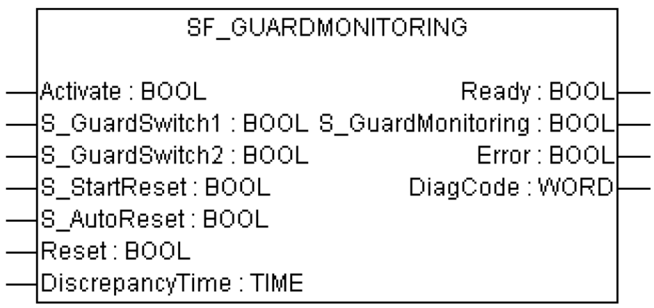
Table 35: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_ESPE_Out = FALSE Error = FALSE
8001	Init	Activation is TRUE. The function block was enabled. Check if S_StartReset is required. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8002	Wait for S_ESPE_In 1	Activation is TRUE. Check if Reset is FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8003	Wait for Reset 1	Activation is TRUE. S_ESPE_In = TRUE. Wait for rising trigger of Reset. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8004	Wait for S_ESPE_In 2	Activation is TRUE. Safety demand detected. Check if Reset is FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8005	Wait for Reset 2	Activation is TRUE. S_ESPE_In = TRUE. Check for S_AutoReset or wait for rising trigger of Reset. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8000	Safety Output Enabled	Activation is TRUE. S_ESPE_In = TRUE. Functional mode with S_ESPE_Out = TRUE. Ready = TRUE S_ESPE_Out = TRUE Error = FALSE

4.6.4.7 SF_GuardMonitoring

Standards	Requirements
EN 953:1997	3.3.3 Control Guard ● The hazardous machine functions "covered" by the guard cannot operate until the guard is closed; ● Closing the guard initiates operation of the hazardous machine function(s).
EN 1088:1995	3.2 Interlocking Guard ● The hazardous machine functions "covered" by the guard cannot operate until the guard is closed; ● If the guard is opened while the hazardous machine functions are operating, a stop instruction is given; ● When the guard is closed, the hazardous machine functions "covered" by the guard can operate, but the closure of the guard does not by itself initiate their operation.
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4 Restart following power failure/spontaneous restart



This function block monitors the relevant safety guard. There are two independent input parameters for two switches at the safety guard coupled with a time difference (MonitoringTime) for closing the guard.

The function block requires two inputs indicating the guard position for safety guards with two switches (according to EN 1088), a DiscrepancyTime input and Reset input. If the safety guard only has one switch, the S_GuardSwitch1 and S_GuardSwitch2 inputs can be bridged. The monitoring time is the maximum time required for both switches to respond when closing the safety guard. The Reset, S_StartReset, and S_AutoReset inputs determine how the function block is reset after the safety guard has been opened.

When opening the safety guard, both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to FALSE. The S_GuardMonitoring output switches to FALSE as soon as one of the switches is set to FALSE. When closing the safety guard, both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to TRUE.

This FB monitors the symmetry of the switching behavior of both switches. The S_GuardMonitoring output remains FALSE if only one of the contacts has completed an open/close process.

The behavior of the S_GuardMonitoring output depends on the time difference between the switching inputs. The discrepancy time is monitored as soon as the value of both S_GuardSwitch1/S_GuardSwitch2 inputs differs. If the DiscrepancyTime has elapsed, but the inputs still differ, the S_GuardMonitoring output remains FALSE. If the second corresponding S_GuardSwitch1/S_GuardSwitch2 input switches to TRUE within the value specified for the DiscrepancyTime input, the S_GuardMonitoring output is set to TRUE following acknowledgment.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 36: FB name: SF_GuardMonitoring

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	↳ *Table 16 “General input parameters” on page 199*
S_GuardSwitch1	BOOL	FALSE	Variable. Guard switch 1 input. FALSE: Guard is open. TRUE: Guard is closed.
S_GuardSwitch2	BOOL	FALSE	Variable. Guard switch 2 input. FALSE: Guard is open. TRUE: Guard is closed.
DiscrepancyTime	TIME	T#0ms	Constant. Configures the monitored synchronous time between S_GuardSwitch1 and S_GuardSwitch2.
S_StartReset	BOOL	FALSE	↳ *Table 16 “General input parameters” on page 199* - only constant
S_AutoReset	BOOL	FALSE	↳ *Table 16 “General input parameters” on page 199* - only constant
Reset	BOOL	FALSE	↳ *Table 16 “General input parameters” on page 199*
VAR_OUTPUT			
Ready	BOOL	FALSE	↳ *Table 17 “General output parameters” on page 200*
S_GuardMonitoring	BOOL	FALSE	Output indicating the status of the guard. FALSE: Guard is not active. TRUE: both S_GuardSwitches are TRUE, no error and acknowledgment. Guard is active.
Error	BOOL	FALSE	↳ *Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	↳ *Table 17 “General output parameters” on page 200*

Typical timing diagrams

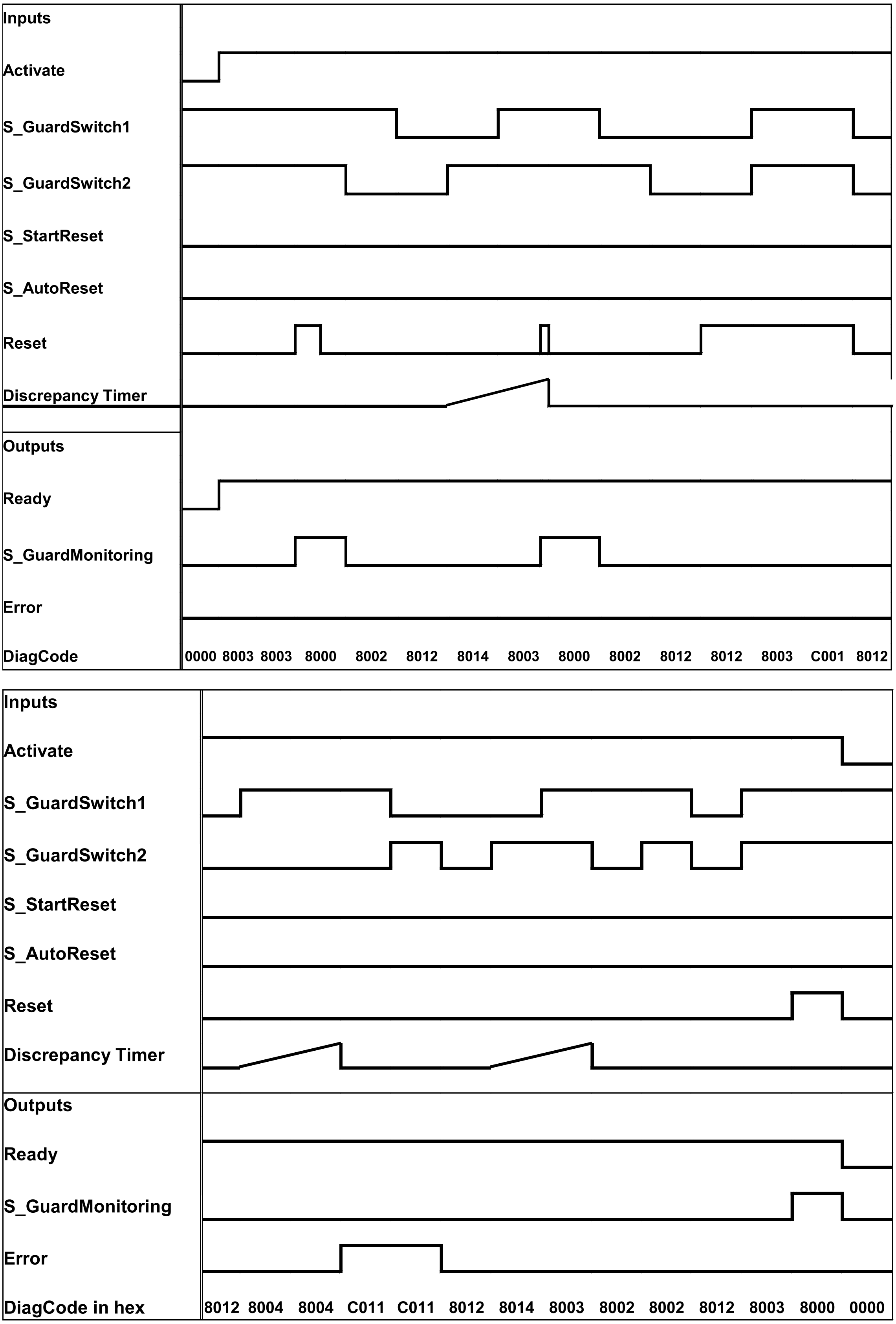


Fig. 103: Timing diagrams for SF_GuardMonitoring

External signals: Mechanical setup combines that of an opening and closing switch according to EN 954 (safety guard with two switches). Discrepancy time monitoring for time lag between both mechanical switches reaction, according to EN 954 (to be considered as "application error" detection, i.e., generated by the application).

An error is detected if the time lag between the first S_GuardSwitch1/S_GuardSwitch2 input and the second is greater than the value for the DiscrepancyTime input. The Error output is set to TRUE.

The function block detects a static TRUE signal at the Reset input.

Error and reset behavior

The S_GuardMonitoring output is set to FALSE. If the two S_GuardSwitch1 and S_Guard-switch2 inputs are bridged, no error is detected. To leave the reset error state, the Reset input must be set to FALSE. To leave the discrepancy time errors, the inputs S_GuardSwitch1 and 2 must both be set to FALSE.

Function block-specific error and status codes

Table 37: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error	Static reset detected in state 8003. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE
C011	Discrepancy-time Error 1	DiscrepancyTime elapsed in state 8004. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE
C012	Discrepancy-time Error 2	DiscrepancyTime elapsed in state 8014. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE

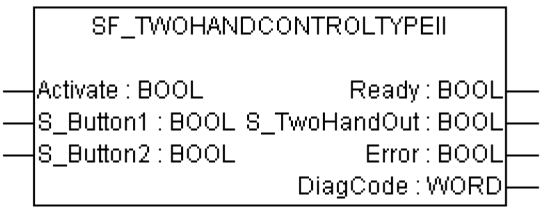
Table 38: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_GuardMonitoring = FALSE Error = FALSE
8000	Normal	Safety guard closed and safe state acknowledged. Ready = TRUE S_GuardMonitoring = TRUE Error = FALSE
8001	Init	Function block has been activated. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8002	Open Guard Request	Complete switching sequence required. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8003	Wait for Reset	Waiting for rising trigger at Reset. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8012	Guard Opened	Guard completely opened. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8004	Wait for Guard-Switch2	S_GuardSwitch1 has been switched to TRUE - waiting for S_GuardSwitch2; discrepancy timer started. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8014	Wait for Guard-Switch1	S_GuardSwitch2 has been switched to TRUE - waiting for S_GuardSwitch1; discrepancy timer started. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8005	Guard Closed	Guard closed. Waiting for Reset, if S_AutoReset = FALSE. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE

4.6.4.8 SF_TwoHandControlTypeII

Standards	Requirements
EN 574:1996	Clause 4, Table 1, Type II. 5.1 Use of both hands / simultaneous actuation. 5.2 Relationship between output signal and input signals. 5.3 Completion of the output signal. 5.6 Reinitiation of the output signal. 6.3 Use of DIN EN 954-1 category 3 (can only be realized by NO and NC switches together with antivalent processing)
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



This function block provides the two-hand control functionality (refer to EN 574, Section 4 Type II).

This function block provides the two-hand control functionality according to EN 574, Section 4 Type II. If S_Button1 and S_Button2 are set to TRUE in a correct sequence, then the S_TwoHandOut output will also be set to TRUE. The FB also controls the release of both buttons before setting the output S_TwoHandOut again to TRUE.

Table 39: FB name: SF_TwoHandControlTypeII

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_Button1	BOOL	FALSE	Variable. Input of button 1 (for category 3 or 4: two antivalent contacts) FALSE: Button 1 released. TRUE: Button 1 actuated.
S_Button2	BOOL	FALSE	Variable. Input of button 2 (for category 3 or 4: two antivalent contacts) FALSE: Button 2 released. TRUE: Button 2 actuated.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_TwoHandOut	BOOL	FALSE	Safety related output signal. FALSE: No correct two hand operation. TRUE: S_Button1 and S_Button2 inputs are TRUE and no error occurred. Correct two hand operation.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Notes: No Reset input or Error output is required, because no test can be performed on both switches.

Typical timing diagram

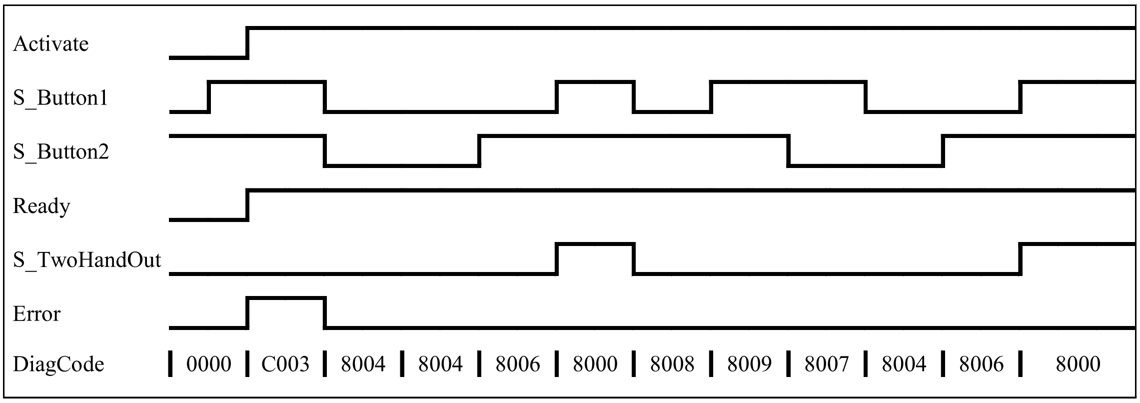


Fig. 104: Timing diagram for SF_TwoHandControlTypeII

After activation of the FB, any button set to TRUE is detected as an invalid input setting leading to an error.

Error behavior

In the event of an error, the S_TwoHandOut output is set to FALSE and remains in this safe state.

The error state is exited when both buttons are released (set to FALSE).

Function block-specific error and status codes

Table 40: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Error B1	S_Button1 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C002	Error B2	S_Button2 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C003	Error B1&B2	The signals at S_Button1 and S_Button2 were TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE

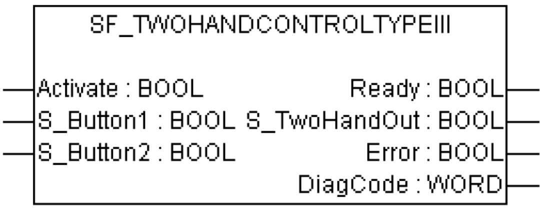
Table 41: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_TwoHandOut = FALSE
8000	Buttons Actuated	Both buttons actuated correctly. The safety related output is enabled. Ready = TRUE Error = FALSE S_TwoHandOut = TRUE
8001	Init	Function block is active, but in the Init state. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8004	Buttons Released	No button is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

DiagCode	State name	State description and output setting
8005	Button 1 Actuated	Only Button 1 is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8006	Button 2 Actuated	Only Button 2 is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8007	Button 2 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is FALSE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8008	Button 1 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is FALSE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8009	Locked Off	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8019	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

4.6.4.9 SF_TwoHandControlTypeIII

Standards	Requirements
EN 574:1996	Clause 4, Table 1, Type III A; B; C. 5.1 Use of both hands / simultaneous actuation. 5.2 Relationship between output signal and input signals. 5.3 Completion of the output signal. 5.6 Reinitiation of the output signal. 5.7 Synchronous actuation. 6.2 Use of DIN EN 954-1 category 1. 6.3 Use of DIN EN 954-1 category 3. (Can only be realized by NO and NC switches together with antivalent processing) 6.4 Use of DIN EN 954-1 category 4. (Can only be realized by NO and NC switches together with antivalent processing)
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



This function block provides the two-hand control functionality (refer to EN 574, Section 4 Type III. Fixed specified time difference is 500 ms).

This function block provides the two-hand control functionality according to EN 574, Section 4 Type III. If S_Button1 and S_Button2 are set to TRUE within 500 ms and in correct sequence, then the S_TwoHandOut output is also set to TRUE. The FB also controls the release of both buttons before setting the output S_TwoHandOut again to TRUE.

Table 42: FB name: SF_TwoHandControlTypeIII

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	➮ *Table 16 “General input parameters” on page 199*
S_Button1	BOOL	FALSE	Variable. Input of button 1 (for category 3 or 4: two antivalent contacts) FALSE: Button 1 released. TRUE: Button 1 actuated.
S_Button2	BOOL	FALSE	Variable. Input of button 2 (for category 3 or 4: two antivalent contacts) FALSE: Button 2 released. TRUE: Button 2 actuated.
VAR_OUTPUT			
Ready	BOOL	FALSE	➮ *Table 17 “General output parameters” on page 200*

Name	Data type	Initial value	Description, parameter values
S_TwoHandOut	BOOL	FALSE	Safety related output signal. FALSE: No correct two hand operation. TRUE: S_Button1 and S_Button2 inputs changed from FALSE to TRUE within 500 ms and no error occurred. The two hand operation has been performed correctly.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Notes: No Reset input or Error output is required, because no test can be performed on both switches.

Typical timing diagram

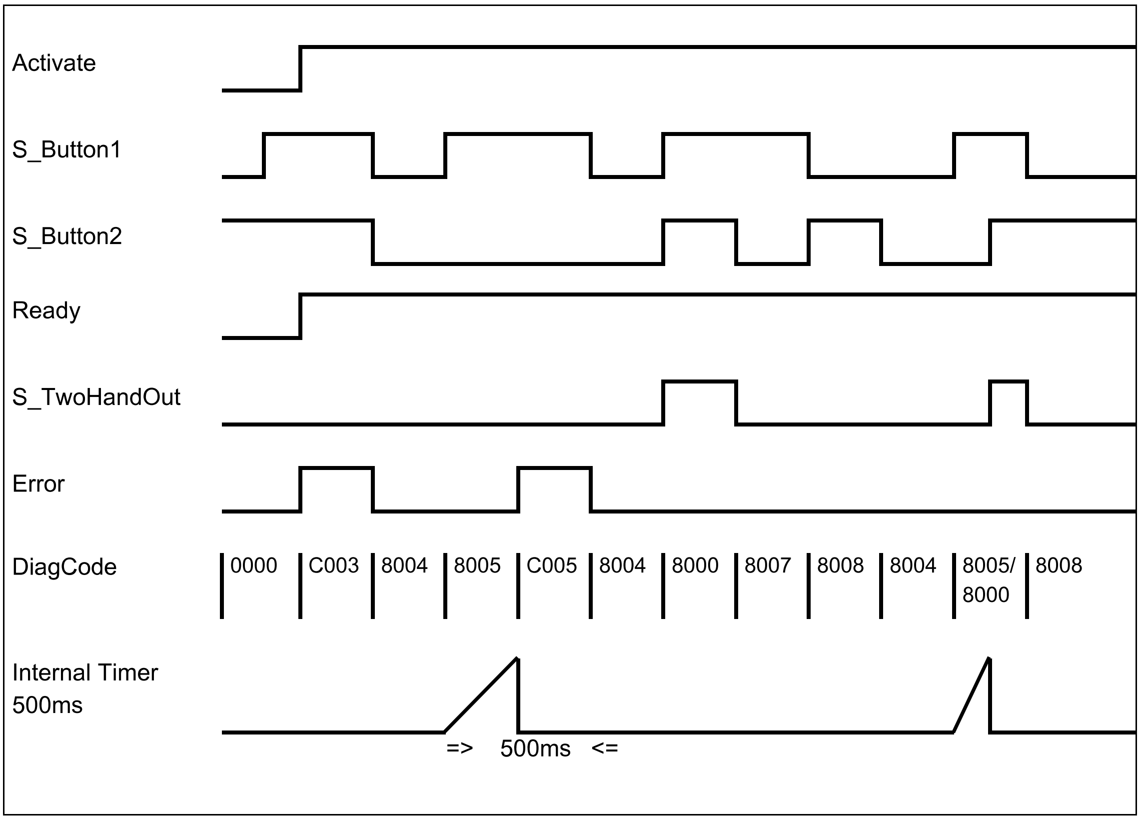


Fig. 105: Timing diagram for SF_TwoHandControlTypeIII

After activation of the FB, any button set to TRUE is detected as an invalid input setting leading to an error. The FB detects when the divergence of the input signals exceeds 500 ms.

Error behavior

In the event of an error, the S_TwoHandOut output is set to FALSE and remains in this safe state.

The error state is exited when both buttons are released (set to FALSE).

Function block-specific error and status codes

Table 43: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Error 1 B1	S_Button1 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C002	Error 1 B2	S_Button2 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C003	Error 1 B1&B2	The signals at S_Button1 and S_Button2 were TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C004	Error 2 B1	S_Button1 was FALSE and S_Button 2 was TRUE after 500 ms in state 8005. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C005	Error 2 B2	S_Button1 was TRUE and S_Button 2 was FALSE after 500 ms in state 8005. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C006	Error 2 B1&B2	S_Button1 was TRUE and S_Button 2 was TRUE after 500 ms in state 8005 or 8006. This state is only possible when the states of the inputs (S_Button1 and S_Button2) change from divergent to convergent (both TRUE) simultaneously when the timer elapses (500 ms) at the same cycle. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE

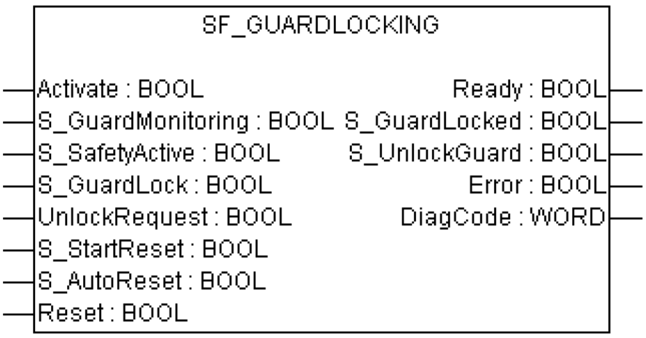
Table 44: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_TwoHandOut = FALSE
8000	Buttons Actuated	Both buttons actuated correctly. The safety related output is enabled. Ready = TRUE Error = FALSE S_TwoHandOut = TRUE
8001	Init	Function block is active, but in the Init state. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8004	Buttons Released	No Button is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8005	Button 1 Actuated	Only Button 1 is actuated. Start monitoring timer. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8006	Button 2 Actuated	Only Button 2 is actuated. Start monitoring timer. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8007	Button 2 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is FALSE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8008	Button 1 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is FALSE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

DiagCode	State name	State description and output setting
8009	Locked Off	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8019	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

4.6.4.10 SF_GuardLocking

Standards	Requirements
EN 953:1997	3.3.3 Control Guard ● The hazardous machine functions "covered" by the guard cannot operate until the guard is closed; ● Closing the guard initiates operation of the hazardous machine function(s).
EN 1088:1995	3.3 Definition: Interlocking Guard With Guard Locking ● The hazardous machine functions "covered" by the guard cannot operate until the guard is closed and locked; ● The guard remains closed and locked until the risk of injury from the hazardous machine functions has passed; ● When the guard is closed and locked, the hazardous machine functions "covered" by the guard can operate, but the closure and locking of the guard do not by themselves initiate their operation. 4.2.2 - Interlocking Device With Guard Locking Conditional unlocking ("four-state interlocking"), refer to Fig. 3 b2 in the standard)
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



This FB controls an entrance to a hazardous area via an interlocking guard with guard locking ("four state interlocking").

The function controls the guard lock and monitors the position of the guard and the lock. This function block can be used with a mechanical locked switch.

The operator requests to get access to the hazardous area. The guard can only be unlocked when the hazardous area is in a safe state. The guard can be locked if the guard is closed. The machine can be started when the guard is closed and the guard is locked. An open guard or unlocked guard will be detected in the event of a safety-critical situation.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 45: FB name: SF_GuardLocking

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
S_GuardMonitoring	BOOL	FALSE	Variable. Monitors the guard interlocking. FALSE: Guard open. TRUE: Guard closed.
S_SafetyActive	BOOL	FALSE	Variable. Status of the hazardous area (EDM), e.g., based on speed monitoring or safe time off delay. FALSE: Machine in "non-safe" state. TRUE: Machine in safe state.
S_GuardLock	BOOL	FALSE	Variable. Status of the mechanical guard locking. FALSE: Guard is not locked. TRUE: Guard is locked.
UnlockRequest	BOOL	FALSE	Variable. Operator intervention - request to unlock the guard. FALSE: No request. TRUE: Request made.
S_StartReset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
S_AutoReset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
Reset	BOOL	FALSE	*Table 16 "General input parameters" on page 199* Also used to request the guard to be locked again. The quality of the signal must conform to a manual reset device (EN 954-1 Ch. 5.4)
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
S_GuardLocked	BOOL	FALSE	Interface to hazardous area which must be stopped. FALSE: No safe state. TRUE: Safe state.
S_UnlockGuard	BOOL	FALSE	Signal to unlock the guard. FALSE: Close guard. TRUE: Unlock guard.

Name	Data type	Initial value	Description, parameter values
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Typical timing diagram

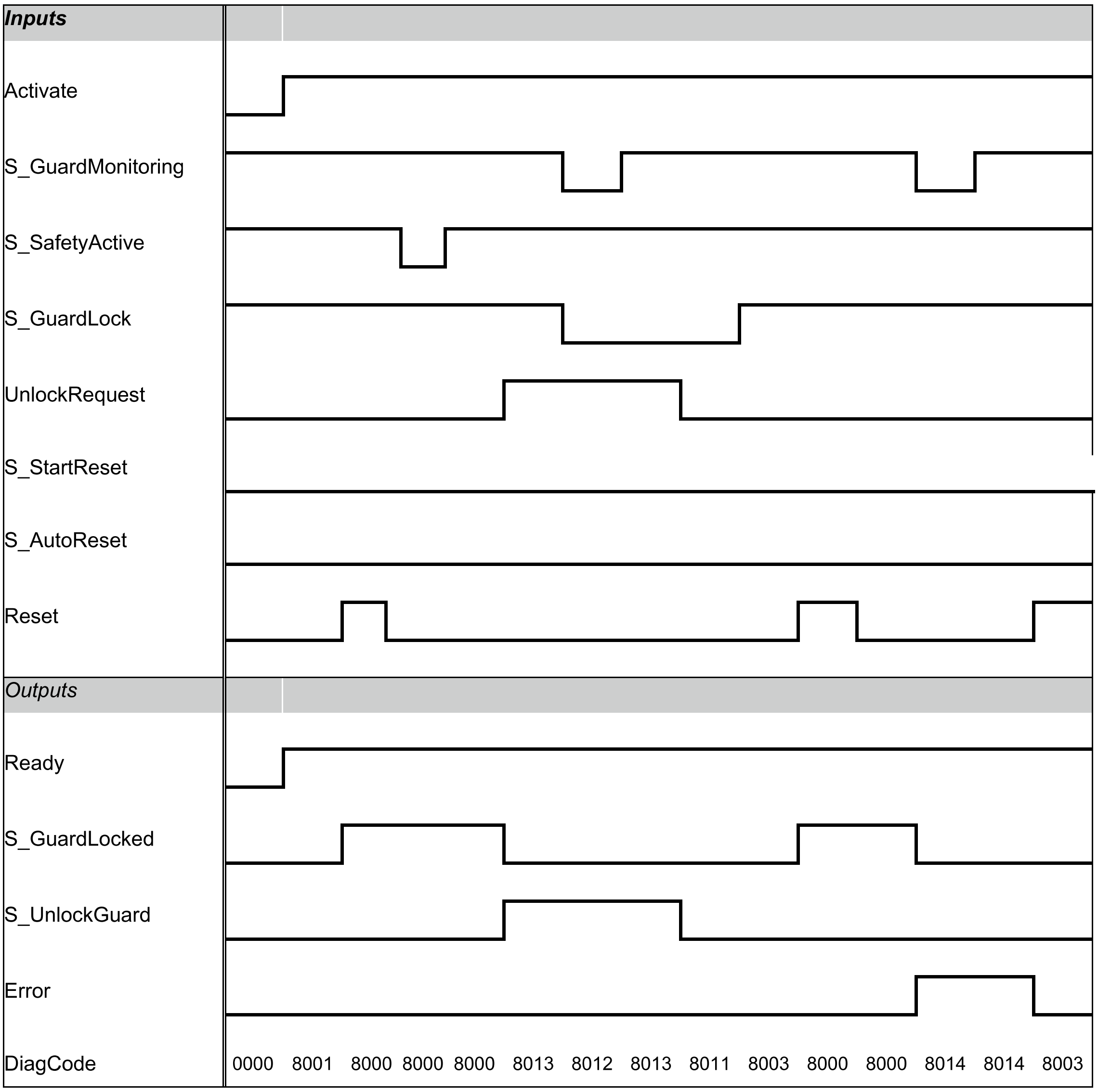


Fig. 106: Timing diagram for SF_GuardLocking

Static signals are detected at Reset. Errors are detected at the guard switches.

Error behavior

In the event of an error the S_GuardLocked and S_UnlockGuard outputs are set to FALSE, the DiagCode output indicates the relevant error code, and the Error output is set to TRUE.

An error must be acknowledged by a rising trigger at the Reset input.

Function block-specific error and status codes

Table 46: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error1	Static Reset detected in state 8001. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C002	Reset Error 2	Static Reset detected in state C004. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C003	Reset Error 3	Static Reset detected in state 8011. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C004	Safety Lost	Safety lost, guard opened or guard unlocked. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE

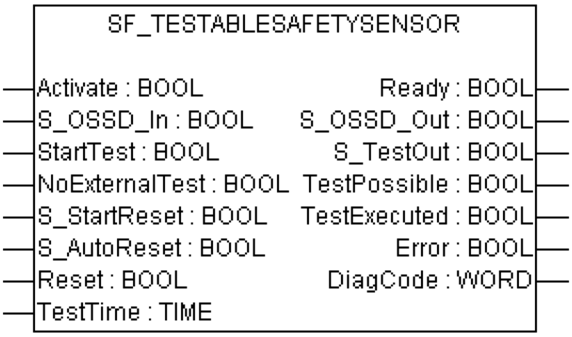
Table 47: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8000	Guard Closed and Locked	Guard is locked. Ready = TRUE S_GuardLocked = TRUE S_UnlockGuard = FALSE Error = FALSE
8001	Init	Function block was activated and initiated. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8003	Wait for Reset	Door is closed and locked, now waiting for operator reset Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8011	Wait for Operator	Waiting for operator to either unlock request or reset. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8012	Guard Open and Unlocked	Lock is released and guard is open. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE Error = FALSE
8013	Guard Closed but Unlocked	Lock is released but guard is closed. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE Error = FALSE
8014	Safety Return	Return of S_SafetyActive signal, now waiting for operator acknowledge. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE

4.6.4.11 SF_TestableSafetySensor

Standards	Requirements
IEC 61496-1:2004	4.2.2.3 Particular requirements for a type 2 ESPE A type 2 ESPE shall have means of periodic test to reveal a failure to danger (for example, loss of detection capability, response time exceeding that specified). A single fault resulting in the loss of detection capability or the increase in response time beyond the specified time or preventing one or more of the OSSDs going to the OFF-state, shall result in a lock-out condition as a result of the next periodic test. Where the periodic test is intended to be initiated by an external (for example, machine) safety-related control system, the ESPE shall be provided with suitable input facilities (for example, terminals). The duration of the periodic test shall be such that the intended safety function is not impaired. Note: If the type 2 ESPE is intended for use as a trip device (for example, when used as a perimeter guard), and the duration of the periodic test is greater than 150 ms, it is possible for a person to pass through the detection zone without being detected. In this case, a restart interlock should be included. If the periodic test is automatically initiated, the correct functioning of the periodic test shall be monitored and a single fault in the parts implementing the monitoring function shall be detected. In the event of a fault, the OSSD(s) shall be signaled to go to the OFF-state. If one or more OSSDs do not go to the OFF-state, a lock-out condition shall be initiated.
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



This function block detects, for example, the loss of the sensing unit detection capability, the response time exceeding that specified, and static ON signal in single-channel sensor systems. It can be used for external testable safety sensors (ESPE: electro-sensitive protective equipment, such as a light beam).

Table 48: FB name: SF_TestableSafetySensor

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_OSSD_In	BOOL	FALSE	Variable. Status of sensor output, e.g., light curtain. FALSE: Safety sensor in test state or demand for safety-related response. TRUE: Sensor in the state for normal operating conditions.

Name	Data type	Initial value	Description, parameter values
StartTest	BOOL	FALSE	Variable. Input to start sensor test. Sets "S_TestOut" and starts the internal time monitoring function in the FB. FALSE: No test requested. TRUE: Test requested.
NoExternalTest	BOOL	FALSE	Constant. Indicates if external manual sensor test is supported. FALSE: The external manual sensor test is supported. Only after a complete manual sensor switching sequence, an automatic test is possible again after a faulty automatic sensor test. TRUE: The external manual sensor test is not supported. An automatic test is possible again without a manual sensor switching sequence after faulty automatic sensor test.
S_StartReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_AutoReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
TestTime	TIME	T#10ms	Constant. Range: 0 ... 150ms. Test time of safety sensor.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_OSSD_Out	BOOL	FALSE	Safety related output indicating the status of the ESPE. FALSE: The sensor has a safety-related action request or test error. TRUE: The sensor has no safety-related action request and no test error.
S_TestOut	BOOL	TRUE	Coupled with the test input of the sensor. FALSE: Test request issued. TRUE: No test request.
TestPossible	BOOL	FALSE	Feedback signal to the process. FALSE: An automatic sensor test is not possible. TRUE: An automatic sensor test is possible.
TestExecuted	BOOL	FALSE	A positive signal edge indicates the successful execution of the automatic sensor test. FALSE: - An automatic sensor test was not executed yet. - An automatic sensor test is active. - An automatic sensor test was faulty. TRUE: A sensor test was executed successfully.

Name	Data type	Initial value	Description, parameter values
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Typical timing diagram

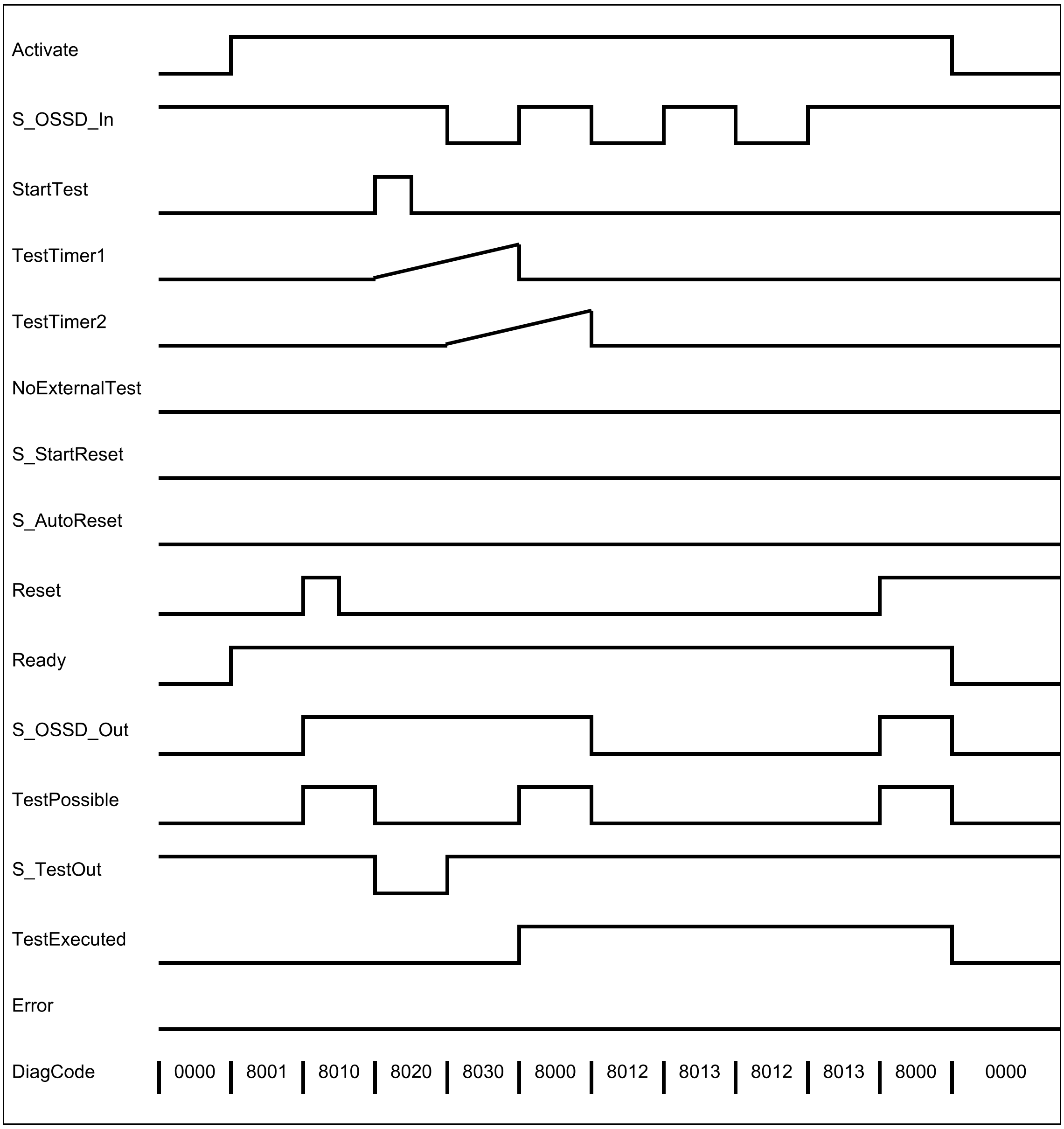


Fig. 107: Timing diagram for SF_TestableSafetySensor

The following conditions force a transition to the error state:

- Test time overrun without delayed sensor feedback.
- Test without sensor signal feedback.
- Invalid static reset signal in the process.
- Plausibility check of the monitoring time setting.

In the event of an error, the S_OSSD_Out output is set to FALSE and remains in this safe state.

Once the error has been removed and the sensor is on (S_OSSD_In = TRUE) - a reset removes the error state and sets the S_OSSD_Out output to TRUE.

If S_AutoReset = FALSE, a rising trigger is required at Reset.

After transition of S_OSSD_In to TRUE, the optional startup inhibit can be reset by a rising edge at the Reset input.

After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

Function block-specific error and status codes

Table 49: FB-specific error codes

DiagCode	State name	State description and output setting
C000	Parameter Error	Invalid value at the TestTime parameter. Values between 0 ms and 150 ms are possible. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C001	Reset Error 1	Static Reset condition detected after FB activation. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C003	Reset Error 3	Static Reset condition detected in state C010. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C004	Reset Error 4	Static Reset condition detected in state C020. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE

DiagCode	State name	State description and output setting
C005	Reset Error 5	Static Reset condition detected in state 8006. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C006	Reset Error 6	Static Reset condition detected in state C000. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C007	Reset Error 7	Static Reset condition detected in state 8013. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = TRUE
C010	Test Error 1	Test time elapsed in state 8020. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C020	Test Error 2	Test time elapsed in state 8030. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE

Table 50: FB-specific status codes (no error):

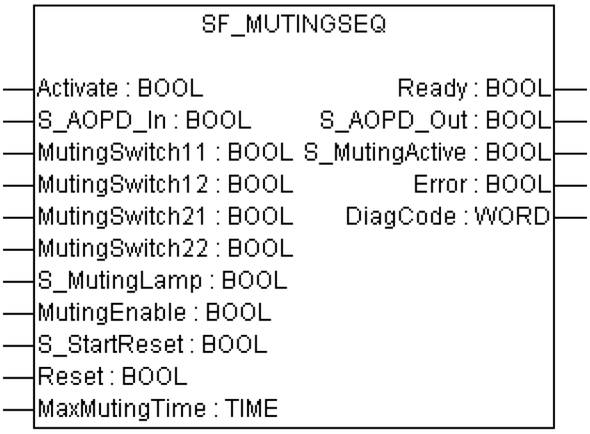
DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8001	Init	An activation has been detected by the FB. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8002	ESPE Interrupted 1	The FB has detected a safety demand. The switch has not been automatically tested yet. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8003	Wait for Reset 1	Wait for rising trigger of Reset after state 8002. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8004	External Function Test	The automatic sensor test was faulty. An external manual sensor test is necessary. The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE). A negative signal edge at the sensor is required. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8005	ESPE Interrupted External Test	The automatic sensor test was faulty. An external manual sensor test is necessary. The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE). A TRUE signal at the sensor is required. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8006	End External Test	The automatic sensor test was faulty. An external manual sensor test is necessary. The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE). The external manual test is complete. The FB detected a complete sensor switching cycle (externally controlled). Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8010	ESPE Free No Test	The FB has not detected a safety demand. The sensor has not been tested automatically. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = FALSE Error = FALSE
8020	Test Request	The automatic sensor test is active. Test Timer is started first time. The transmitter signal of the sensor is switched off by the FB. The signal of the receiver must follow the signal of the transmitter. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = FALSE TestPossible = FALSE TestExecuted = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8030	Test Active	The automatic sensor test is active. Test Timer is started second time. The transmitter signal of the sensor is switched on by the FB. The signal of the receiver must follow the signal of the transmitter. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8000	ESPE Free Test ok	The FB has not detected a safety demand. The sensor was automatically tested. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = TRUE Error = FALSE
8012	ESPE Inter-rupted 2	The FB has detected a safety demand. The switch was automatically tested. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = FALSE
8013	Wait for Reset 2	Wait for rising trigger of Reset after state 8012. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = FALSE

4.6.4.12 SF_MutingSeq

Standards	Requirements
IEC 61496-1:2004	A.7 Muting A.7.1.2 There shall be at least two independent hard-wired muting signal sources to initiate the function. It shall not be possible to initiate muting when the OSSDs are already in the OFF-state. A.7.1.3 The mute function shall only be initiated by the correct sequence and/or timing of the mute signals. Should conflicting muting signals occur, the ESPE shall not allow a muted condition to occur. A.7.1.4 There shall be at least two independent hard-wired muting signal sources to stop the function. The muting function shall stop when the first of these muting signals changes state. The deactivation of the muting function shall not rely only on the clearance of the ESPE. A.7.1.5 The muting signals should be continuously present during muting. When the signals are not continuously present, an incorrect sequence and/or the expiration of a pre-set time limit shall cause either a lock-out condition or a restart interlock. A.7.4 Indication: A mute status signal or indicator shall be provided (in some applications, an indication signal of muting is necessary)
IEC 62046/ Ed. 1:2005	5.5.1: ... an indicator to show when the muting function is active can be necessary. The muting function shall be initiated and terminated automatically ... Incorrect signals, sequence, or timing of the muting sensors or signals shall not allow a mute condition. It shall not be possible to initiate the muting function when: ● the protective equipment OSSDs are in the OFF-state; ● the protective equipment is in the lock-out condition. ● initiation of the muting function by two or more independent muting sensors such that a single fault cannot cause a muted condition; ● termination of the muting function by two or more independent muting sensors such that deactivation of one sensor will terminate the muting function; ● use of timing and sequence control of the muting sensors to ensure correct muting operation. 5.5.3: The following measures shall be considered: ... ● limiting muting to a fixed time that is only sufficient for the material to pass through the detection zone. When this time is exceeded, the muting function should be canceled and all hazardous movements stopped. Annex F.3 Four beams - sequence control: (refer also to Fig. F.3.1 and table F.1 in the standard) The initiation of the muting function depends on monitoring the correct sequence of activation of the muting sensors. For example, in the muted condition, if S2 (in this document MS_12) is deactivated before S3 (in this document MS_21) is activated, muting is terminated. Annex F.5: Methods to avoid manipulation of the muting function: ... use a muting enable command generated by the control system of the machine that will only enable the muting function when needed by the machine cycle.
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



Muting is the intended suppression of the safety function (e.g., light barriers). In this FB, sequential muting with four muting sensors is specified.

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be proximity switches, photoelectric barriers, limit switches, etc. which do not have to be fail-safe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, sequential muting with four muting sensors was used; an explanation for the forward direction of transportation is provided below. The FB can be used in both directions, forward and backward. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation. When the MutingEnable signal is not available, this input must be set to TRUE.

The FB input parameters include the signals of the four muting sensors (MutingSwitch11 ... MutingSwitch22) as well as the OSSD signal from the AOPD device (S_AOPD_In).

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 51: FB name: SF_MutingSeq

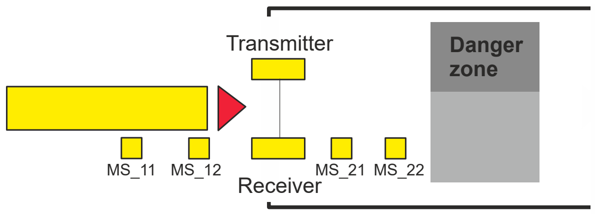
Name	**Data type**	**Initial value**	**Description, parameter values**
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_AOPD_In	BOOL	FALSE	Variable. OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
MutingSwitch11	BOOL	FALSE	Variable. Status of muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11.
MutingSwitch12	BOOL	FALSE	Variable. Status of muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12.

Name	Data type	Initial value	Description, parameter values
MutingSwitch21	BOOL	FALSE	Variable. Status of muting sensor 21. FALSE: Muting sensor 21 not actuated. TRUE: Workpiece actuates muting sensor 21.
MutingSwitch22	BOOL	FALSE	Variable. Status of muting sensor 22. FALSE: Muting sensor 22 not actuated. TRUE: Workpiece actuates muting sensor 22.
S_MutingLamp	BOOL	FALSE	Variable or constant. Indicates operation of the muting lamp. FALSE: Muting lamp failure. TRUE: No muting lamp failure
MutingEnable	BOOL	FALSE	Variable or constant. Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of muting function enabled
S_StartReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
MaxMutingTime	TIME	T#0s	Constant 0 .. 10 min; Maximum time for complete muting sequence, timer started when first muting sensor is actuated.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_AOPD_Out	BOOL	FALSE	Safety related output, indicates status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
S_MutingActive	BOOL	FALSE	Indicates status of Muting process. FALSE: Muting not active. TRUE: Muting active.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Note: A short circuit in the muting sensor signals or a functional application error to supply these signals is not detected by this FB but interpreted as incorrect muting sequence. However, this condition should not lead to unwanted muting. The user should take care to include this in his risk analysis.

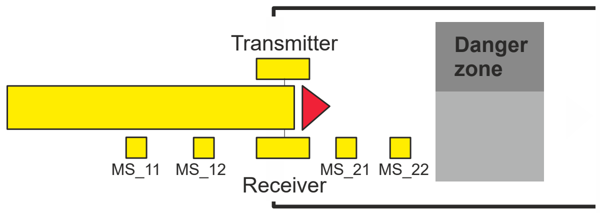
Example for SF_MutingSeq in forward direction with four sensors

1



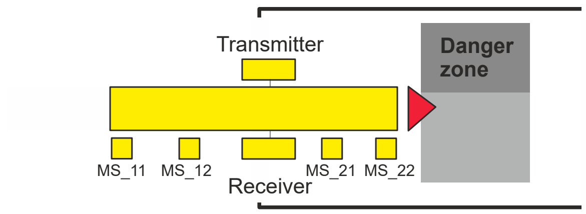
If muting sensor MutingSwitch12 (MS_12) is activated by the product after MutingSwitch11 (MS_11), the muting mode is activated.

2



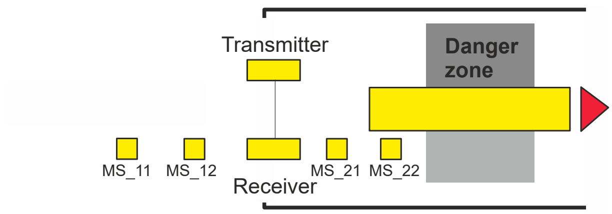
Muting mode remains active as long as MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product. The product may pass through the light curtain without causing a machine stop.

3



Before muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are disabled, muting sensors MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be activated. This ensures that muting mode remains active.

4



Muting mode is terminated if only muting sensor MutingSwitch22 (MS_22) is activated by the product.

Muting conditions

Forward direction

Muting condition 1 (to state 8011) (MS_11 is the first actuated entry switch). Start timer MaxMutingTime:

MutingEnable AND (R_TRIG at MS_11 AND NOT MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 2 (from state 8011 to state 8012) (MS_12 is the second actuated entry switch):

MutingEnable AND (MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 3 (from state 8012 to state 8000) (MS_21 is the first released exit switch). Stop timer MaxMutingTime:

NOT MS_11 AND NOT MS_12 AND F_TRIG at MS_21 AND MS_22

Backward direction

Muting condition 11 (to state 8122) (MS_22 is the first actuated entry switch). Start timer MaxMutingTime:

MutingEnable AND (NOT MS_11 AND NOT MS_12 AND NOT MS_21 AND R_TRIG at MS_22)

Muting condition 12 (from state 8122 to state 8112) (MS_21 is the second actuated entry switch):

MutingEnable AND (NOT MS_11 AND NOT MS_12 AND R_TRIG at MS_21 AND MS_22)

Muting condition 13 (MS_12 is the first released exit switch). Stop timer MaxMutingTime:

MS_11 AND F_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22

Specification of wrong muting sequences:

In state 8000 - (NOT MutingEnable AND R_TRIG at MS_11) OR (NOT MutingEnable AND R_TRIG at MS_22) OR (MS_12 OR MS_21) OR (MS_11 AND MS_22)

In state 8011 - NOT MutingEnable OR NOT MS_11 OR MS_21 OR MS_22

In state 8012 - R_TRIG at MS_11 OR R_TRIG at MS_12 OR F_TRIG at MS_22

In state 8122 - NOT MutingEnable OR MS_11 OR MS_12 OR NOT MS_22

In state 8112 - F_TRIG at MS_11 OR R_TRIG at MS_21 OR R_TRIG at MS_22

Typical timing diagram

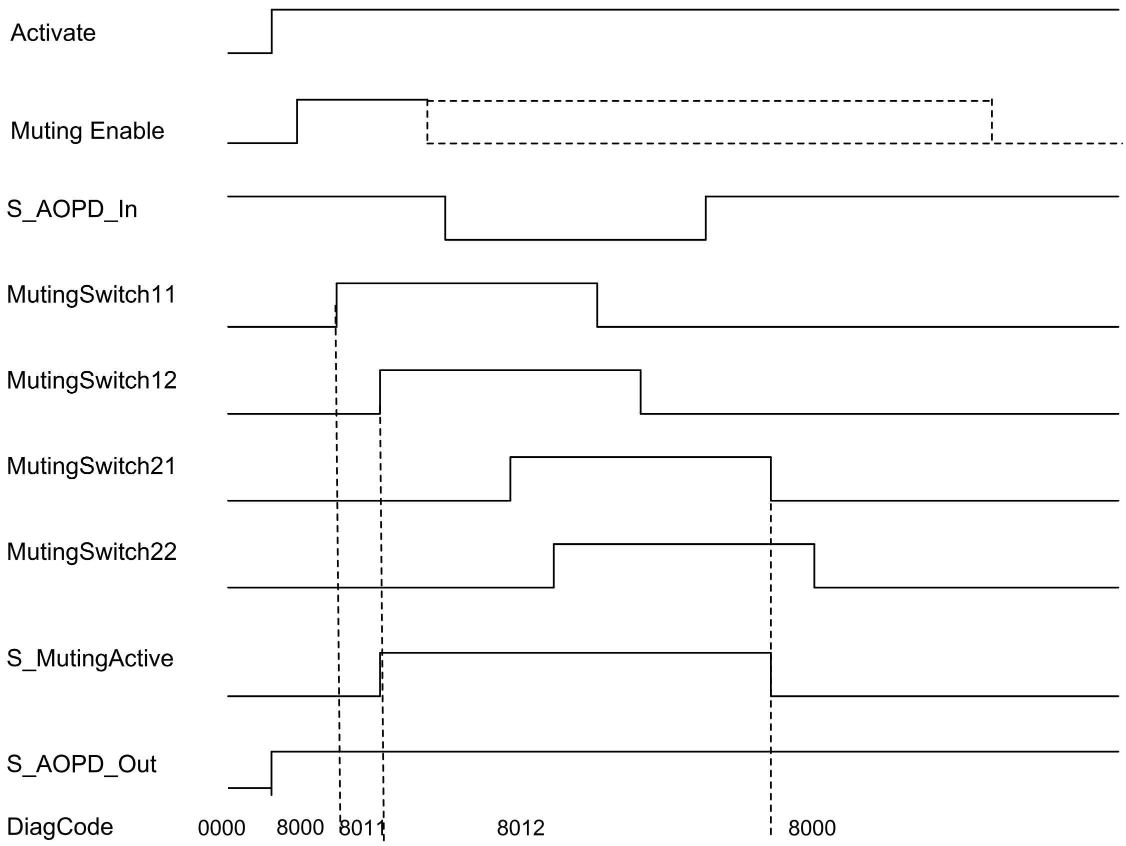


Fig. 108: Timing diagram for SF_MutingSeq with S_StartReset = TRUE

The FB detects the following error conditions:

- Muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable.
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.

- A static Reset condition.
- MaxMutingTime has been set to a value less than T#0s or greater than T#10min.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.

Error behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.

A restart is inhibited until the error conditions are cleared and the safe state is acknowledged with Reset by the operator.

Function block-specific error and status codes

Table 52: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error 1	Static Reset condition detected after FB activation. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C003	Error Muting lamp	Error detected in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
CYx4	Error Muting sequence	Error detected in muting sequence in states 8000, 8011, 8012, 8112 or 8122. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence (2 states for forward and 2 states for backward direction). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state Forward 8011 C2x4 = Error occurred in state Forward 8012 C3x4 = Error occurred in state Backward 8122 C4x4 = Error occurred in state Backward 8112 CFx4 = Muting enable missing x = Status of the sensors when error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).

DiagCode	State name	State description and output setting
C005	Parameter Error	MaxMutingTime value out of range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error Timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

Table 53: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety demand from AOPD. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block has been activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety demand detected by AOPD, muting not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Forward Start	Muting forward, sequence is in starting phase and no safety demand. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8012	Muting Forward Active	Muting forward, sequence is active. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8112	Muting Backward Active	Muting backward, sequence is active. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8122	Muting Backward Start	Muting backward, sequence is in starting phase and no safety demand. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

4.6.4.13 SF_MutingPar

<table>
<tr><th>Standards</th><th>Requirements</th></tr>
<tr><td>IEC 61496-1:2004</td><td>A.7 Muting

A.7.1.2 There shall be at least two independent hard-wired muting signal sources to initiate the function. It shall not be possible to initiate muting when the OSSDs are already in the OFF-state.

A.7.1.3 The mute function shall only be initiated by the correct sequence and/or timing of the mute signals. Should conflicting muting signals occur, the ESPE shall not allow a muted condition to occur.

A.7.1.4 There shall be at least two independent hard-wired muting signal sources to stop the function. The muting function shall stop when the first of these muting signals changes state. The deactivation of the muting function shall not rely only on the clearance of the ESPE.

A.7.1.5 The muting signals should be continuously present during muting. When the signals are not continuously present, an incorrect sequence and/or the expiration of a pre-set time limit shall cause either a lock-out condition or a restart interlock.

A.7.4 Indication: A mute status signal or indicator shall be provided (in some applications, an indication signal of muting is necessary.</td></tr>
<tr><td>IEC 62046/
Ed. 1:2005</td><td>5.5.1: ... an indicator can be necessary to show when the muting function is active.

The muting function shall be initiated and terminated automatically ... Incorrect signals, sequence, or timing of the muting sensors or signals shall not allow a mute condition. It shall not be possible to initiate the muting function when:

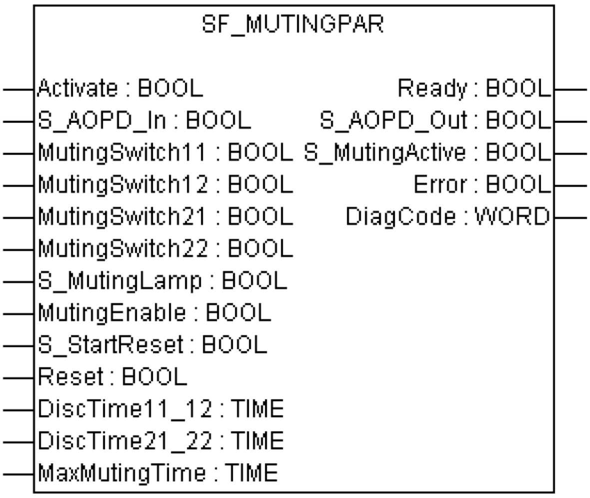
● the protective equipment OSSDs are in the OFF-state;
● the protective equipment is in the lock-out condition;
● initiation of the muting function by two or more independent muting sensors such that a single fault cannot cause a muted condition;
● termination of the muting function by two or more independent muting sensors such that deactivation of one sensor will terminate the muting function;
● use of timing and sequence control of the muting sensors to ensure correct muting operation.

5.5.3: The following measures shall be considered: ...

● limiting muting to a fixed time that is only sufficient for the material to pass through the detection zone. When this time is exceeded, the muting function should be canceled and all hazardous movements stopped;

Annex F.2 Four beams - timing control: (refer also to Fig. F.2.4 in the standard): The monitoring of the muting function is based on time limitation between the actuation of the sensors S1 (in this document MS_11) and S2 (in this document MS_12) and between the actuation of sensors S3 (in this document MS_21) and S4 (in this document MS_22). A maximum time limit of 4 s is recommended. The muting function is initiated by the two sensors S1, S2 and maintained by the two sensors S3, S4; this means that for a certain time all the four sensors are activated. The muting function is terminated when S3 or S4 is deactivated.

Annex F.5: Methods to avoid manipulation of the muting function: ... use a muting enable command generated by the control system of the machine that will only enable the muting function when needed by the machine cycle.</td></tr>
<tr><td>EN 954-1:1996</td><td>5.4 Manual reset</td></tr>
<tr><td>ISO 12100-2:2003</td><td>4.11.4: Restart following power failure/spontaneous restart</td></tr>
</table>



Muting is the intended suppression of the safety function. In this FB, parallel muting with four muting sensors is specified.

This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be proximity switches, photo-electric barriers, limit switches, etc. which do not have to be fail-safe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, parallel muting with four muting sensors was used; an explanation is provided below. The FB can be used in both directions, forward and backward. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation.

The FB input parameters include the signals of the four muting sensors (MutingSwitch11 .. MutingSwitch22), the OSSD signal from the AOPD device (S_AOPD_In) as well as three parameterizable times (DiscTime11_12, DiscTime21_22 and MaxMutingTime).

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 54: FB name: SF_MutingPar

Name	**Data type**	**Initial value**	**Description, parameter values**
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_AOPD_In	BOOL	FALSE	Variable. OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
MutingSwitch11	BOOL	FALSE	Variable. Status of muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11.
MutingSwitch12	BOOL	FALSE	Variable. Status of muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12.

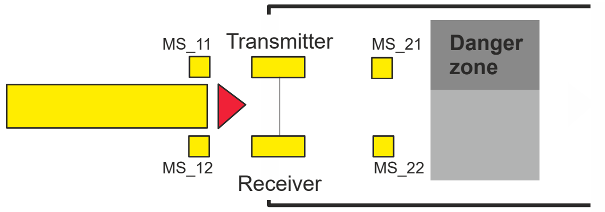
Name	Data type	Initial value	Description, parameter values
MutingSwitch21	BOOL	FALSE	Variable. Status of muting sensor 21. FALSE: Muting sensor 21 not actuated. TRUE: Workpiece actuates muting sensor 21.
MutingSwitch22	BOOL	FALSE	Variable. Status of muting sensor 22. FALSE: Muting sensor 22 not actuated. TRUE: Workpiece actuates muting sensor 22.
S_MutingLamp	BOOL	FALSE	Variable or constant. Indicates operation of the muting lamp. FALSE: Muting lamp failure. TRUE: No muting lamp failure.
MutingEnable	BOOL	FALSE	Variable or constant. Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of muting function enabled
S_StartReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
DiscTime11_12	TIME	T#0s	Constant 0..4 s; Maximum discrepancy time for MutingSwitch11 and MutingSwitch12.
DiscTime21_22	TIME	T#0s	Constant 0..4 s; Maximum discrepancy time for MutingSwitch21 and MutingSwitch22.
MaxMutingTime	TIME	T#0s	Constant 0..10 min; Maximum time for complete muting sequence, timer started when first muting sensor is actuated.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_AOPD_Out	BOOL	FALSE	Safety related output, indicates status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
S_MutingActive	BOOL	FALSE	Indicates status of muting process. FALSE: Muting not active. TRUE: Muting active.

Name	Data type	Initial value	Description, parameter values
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Note: A short circuit in the muting sensor signals or a functional application error to supply these signals is not detected by this FB. However, this condition should not lead to unwanted muting. The user should take care to include this in his risk analysis.

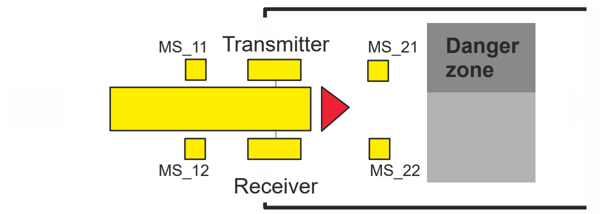
Example for SF_MutingPar in forward direction with four sensors

1



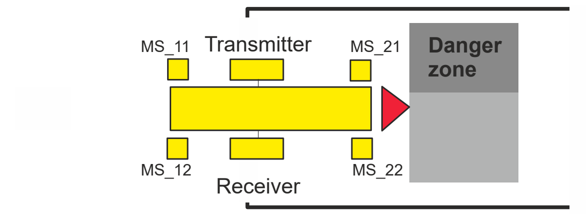
If the muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product within the time DiscTime11_12, muting mode is activated (S_MutingActive = TRUE).

2



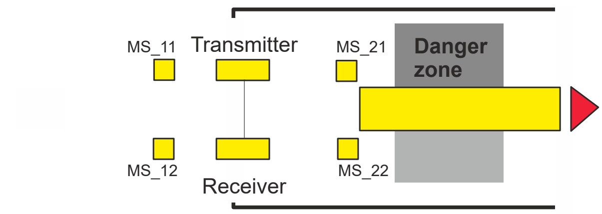
Muting mode remains active as long as MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product. The product may pass through the light curtain without causing a machine stop.

3



Before muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are disabled, muting sensors MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be activated. This ensures that muting mode remains active. The time discrepancy between switching of MutingSwitch21 and MutingSwitch22 is monitored by the time DiscTime21_22.

4



Muting mode is terminated if either muting sensor MutingSwitch21 (MS_21) or MutingSwitch22 (MS_22) is disabled by the product. The maximum time for muting mode to be active is the MaxMutingTime.

Muting conditions

Forward direction

Muting condition 1 (to state 8011) (MS_11 is the first actuated entry switch). Start timers MaxMutingTime and DiscTime11_12:

MutingEnable AND (R_TRIG at MS_11 AND NOT MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 1 (to state 8311) (MS_12 is the first actuated entry switch). Start timers MaxMutingTime and DiscTime11_12:

MutingEnable AND (NOT MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 2 (from state 8011) (MS_12 is the second actuated entry switch). Stop timer DiscTime11_12:

MutingEnable AND (MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 2 (from state 8311) (MS_11 is the second actuated entry switch). Stop timer DiscTime11_12:

MutingEnable AND (R_TRIG at MS_11 AND MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 3 (both entry switches are actuated in same cycle). Start timer MaxMutingTime:

MutingEnable AND (R_TRIG at MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 4 (all switches are actuated): MS_11 AND MS_12 AND MS_21 AND MS_22

Muting condition 24 (to state 8014) (MS_21 is the first actuated exit switch). Start timer DiscTime21_22:

MS_11 AND MS_12 AND R_TRIG at MS_21 AND NOT MS_22

Muting condition 24 (to state 8314) (MS_22 is the first actuated exit switch). Start timer DiscTime21_22:

MS_11 AND MS_12 AND NOT MS_21 AND R_TRIG at MS_22

Muting condition 25 (from state 8014) (MS_22 is the second actuated exit switch). Stop timer DiscTime21_22:

MS_11 AND MS_12 AND MS_21 AND R_TRIG at MS_22

Muting condition 25 (from state 8314) (MS_21 is the second actuated exit switch). Stop timer DiscTime21_22:

MS_11 AND MS_12 AND R_TRIG at MS_21 AND MS_22

Muting condition 5 (one of the exit is switches is released). Stop timer MaxMutingTime:

NOT MS_11 AND NOT MS_12 AND (F_TRIG at MS_21 OR F_TRIG at MS_22)

Backward direction

Muting condition 11 (to state 8122) (MS_21 is the first actuated entry switch). Start timers MaxMutingTime and DiscTime21_22:

MutingEnable AND (NOT MS_22 AND R_TRIG at MS_21 AND NOT MS_11 AND NOT MS_12)

Muting condition 11 (to state 8422) (MS_22 is the first actuated entry switch). Start timers MaxMutingTime and DiscTime21_22:

MutingEnable AND (R_TRIG at MS_22 AND NOT MS_21 AND NOT MS_11 AND NOT MS_12)

Muting condition 12 (from state 8122) (MS_22 is the second actuated entry switch). Stop timer DiscTime21_22:

MutingEnable AND (MS_21 AND R_TRIG at MS_22 AND NOT MS_11 AND NOT MS_12)

Muting condition 12 (from state 8422) (MS_21 is the second actuated entry switch). Stop timer DiscTime21_22:

MutingEnable AND (R_TRIG at MS_21 AND MS_22 AND NOT MS_11 AND NOT MS_12)

Muting condition 13 (both entry switches are actuated in same cycle). Start timer MaxMutingTime:

MutingEnable AND (R_TRIG at MS_21 AND R_TRIG at MS_22 AND NOT MS_11 AND NOT MS_12)

Muting condition 14 (all switches are actuated): MS_11 AND MS_12 AND MS_21 AND MS_22

Muting condition 44 (to state 8114) (MS_11 is the first actuated exit switch). Start timer DiscTime11_12:

MS_21 AND MS_22 AND R_TRIG at MS_11 AND NOT MS_12

Muting condition 44 (to state 8414) (MS_12 is the first actuated exit switch). Start timer DiscTime11_12:

MS_21 AND MS_22 AND NOT MS_11 AND R_TRIG at MS_12

Muting condition 45 (from state 8114) (MS_12 is the second actuated exit switch). Stop timer DiscTime11_12:

MS_21 AND MS_22 AND MS_11 AND R_TRIG at MS_12

Muting condition 45 (from state 8414) (MS_11 is the second actuated exit switch). Stop timer DiscTime11_12:

MS_21 AND MS_22 AND R_TRIG at MS_11 AND MS_12

Muting condition 15 (one of the exit switches is released). Stop timer MaxMutingTime:

NOT MS_21 AND NOT MS_22 AND (F_TRIG at MS_11 OR F_TRIG at MS_12)

Wrong muting sequences:

State 8000 - (MutingEnable = FALSE when muting sequence starts) OR

((MS_11 OR MS_12) AND (MS_21 OR MS_22)) OR

(R_TRIG at MS_11 AND MS_12 AND NOT R_TRIG at MS_12) OR

(R_TRIG at MS_12 AND MS_11 AND NOT R_TRIG at MS_11) OR

(R_TRIG at MS_21 AND MS_22 AND NOT R_TRIG at MS_22) OR

(R_TRIG at MS_22 AND MS_21 AND NOT R_TRIG at MS_21) OR

((MS_11 AND NOT R_TRIG at MS_11) AND (MS_12 AND NOT R_TRIG at MS_12)) OR

((MS_21 AND NOT R_TRIG at MS_21) AND (MS_22 AND NOT R_TRIG at MS_22))

State 8011 - NOT MutingEnable OR NOT MS_11 OR MS_21 OR MS_22

State 8311 - NOT MutingEnable OR NOT MS_12 OR MS_21 OR MS_22

State 8012 - NOT MS_11 OR NOT MS_12

State 8021 - R_TRIG at MS_11 OR R_TRIG at MS_12 OR R_TRIG at MS_21 OR R_TRIG at MS_22

State 8014 - NOT MS_11 OR NOT MS_12 OR NOT MS_21

State 8314 - NOT MS_11 OR NOT MS_12 OR NOT MS_22

State 8122 - NOT MutingEnable OR MS_11 OR MS_12 OR NOT MS_21

State 8422 - NOT MutingEnable OR MS_11 OR MS_12 OR NOT MS_22

State 8121 - NOT MS_21 OR NOT MS_22

State 8112 - R_TRIG at MS_11 OR R_TRIG at MS_12 OR R_TRIG at MS_21 OR R_TRIG at MS_22

State 8114 - NOT MS_21 OR NOT MS_22 OR NOT MS_11

State 8414 - NOT MS_21 OR NOT MS_22 OR NOT MS_12

Typical timing diagram

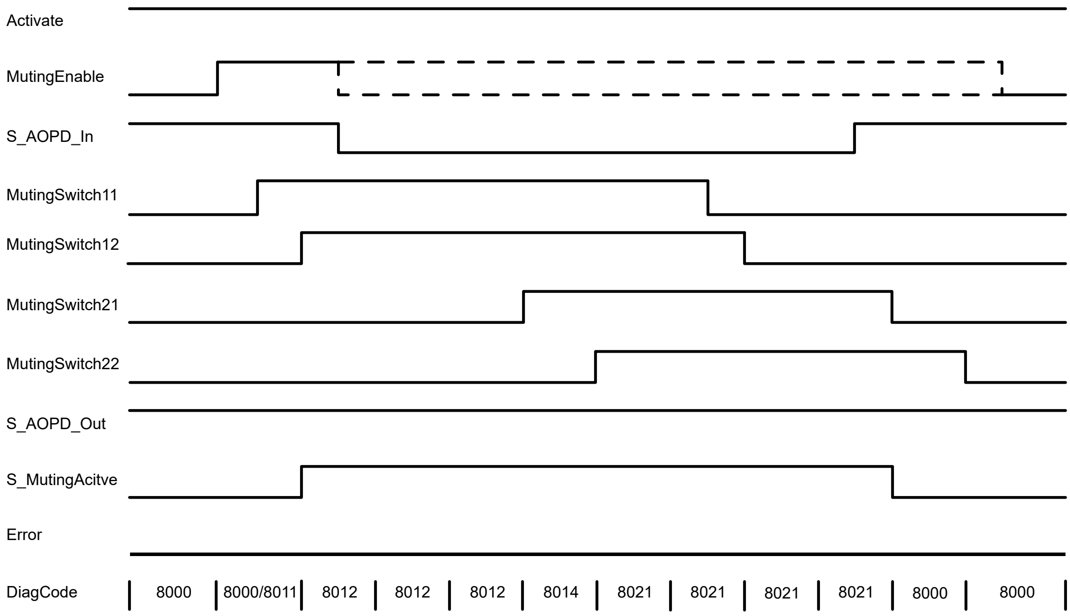


Fig. 109: Timing diagram for SF_MutingPar

The FB detects the following error conditions:

- DiscTime11_12 and DiscTime21_22 have been set to values less than T#0s or greater than T#4s.
- MaxMutingTime has been set to a value less than T#0s or greater than T#10min.
- The discrepancy time for the MutingSwitch11/MutingSwitch12 or MutingSwitch21/MutingSwitch22 sensor pairs has been exceeded.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable.
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static Reset condition is detected in states 8001 and 8003.

Error behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.

A restart is inhibited until the error conditions are cleared and the safe state is acknowledged with Reset by the operator.

Function block-specific error and status codes

Table 55: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error 1	Static Reset condition detected after FB activation in state 8001. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C003	Error Muting Lamp	Error detected in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
CYx4	Error Muting sequence	Error detected in muting sequence state 8000, 8011, 8311, 8012, 8021, 8014, 8314, 8122, 8422, 8121, 8112, 8114 or 8414. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence (6 states for forward and 6 states for backward direction). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state Forward 8011 C2x4 = Error occurred in state Forward 8311 C3x4 = Error occurred in state Forward 8012 C4x4 = Error occurred in state Forward 8014 C5x4 = Error occurred in state Forward 8314 C6x4 = Error occurred in state Forward 8021 C7x4 = Error occurred in state Backward 8122 C8x4 = Error occurred in state Backward 8422 C9x4 = Error occurred in state Backward 8121 CAx4 = Error occurred in state Backward 8114 CBx4 = Error occurred in state Backward 8414 CCx4 = Error occurred in state Backward 8112 ... CFx4 = Muting enable missing x = Status of sensors when error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).

DiagCode	State name	State description and output setting
C005	Parameter Error	DiscTime11_12, DiscTime21_22 or MaxMutingTime value out of range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error Timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C007	Error Timer MS11_12	Timing error: Discrepancy time for switching MutingSwitch11 and MutingSwitch12 > DiscTime11_12. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C008	Error Timer MS21_22	Timing error: Discrepancy time for switching MutingSwitch21 and MutingSwitch22 > DiscTime21_22. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

Table 56: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety demand from AOPD. If timers from subsequent muting are still running, they are stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

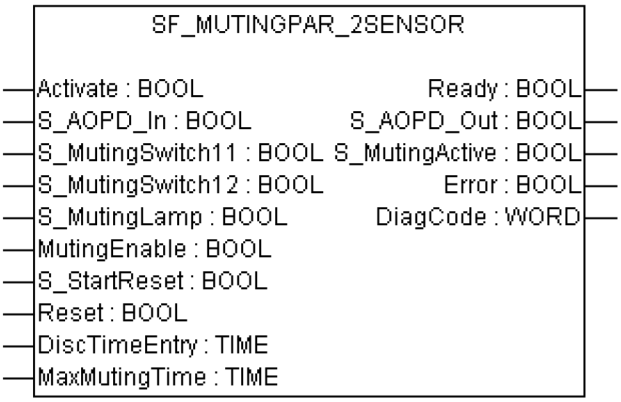
DiagCode	State name	State description and output setting
8001	Init	Function block has been activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety demand detected by AOPD, muting not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Forward Start 1	Muting forward sequence is in starting phase after rising trigger of MutingSwitch 11. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8311	Muting Forward Start 2	Muting forward sequence is in starting phase after rising trigger of MutingSwitch 12. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8012	Muting Forward Active 1	Muting forward sequence is active either: - After rising trigger of the second entry MutingSwitch 12 or 11 has been detected. - When both MutingSwitch 11 and 12 have been actuated in the same cycle. Monitoring of DiscTime11_12 is stopped. Monitoring of MaxMutingTime is activated, when transition came directly from state 8000. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8014	Muting Forward Step 1	Muting forward sequence is active. MutingSwitch21 is the first actuated exit switch. Monitoring of DiscTime21_22 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8314	Muting Forward Step 2	Muting forward sequence is active. MutingSwitch22 is the first actuated exit switch. Monitoring of DiscTime21_22 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8021	Muting Forward Active 2	Muting forward sequence is still active. Both MutingSwitch21 and 22 are actuated, the monitoring of DiscTime21_22 is stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8122	Muting Backward Start 1	Muting backward sequence is in starting phase after rising trigger of MutingSwitch21. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8422	Muting Backward Start 2	Muting backward sequence is in starting phase after rising trigger of MutingSwitch22. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8121	Muting Back-ward Active 1	Muting backward sequence is active either: - After rising trigger of the second entry MutingSwitch 21 or 22 has been detected. - When both MutingSwitch 21 and 22 have been actuated in the same cycle. Monitoring of DiscTime21_22 is stopped. Monitoring of MaxMutingTime is activated, when transition came directly from state 8000. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8114	Muting Back-ward Step 1	Muting backward sequence is active. MutingSwitch11 is the first actuated exit switch. Monitoring of DiscTime11_12 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8414	Muting Back-ward Step 2	Muting backward sequence is active. MutingSwitch12 is the first actuated exit switch. Monitoring of DiscTime11_12 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8112	Muting Back-ward Active 2	Muting backward sequence is still active. Both exit switches MutingSwitch11 and 12 are actuated, the monitoring of Dis-cTime11_12 is stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE

4.6.4.14 SF_MutingPar_2Sensor

Standards	Requirements
IEC 61496-1:2004	A.7 Muting A.7.1.2 There shall be at least two independent hard-wired muting signal sources to initiate the function. It shall not be possible to initiate muting when the OSSDs are already in the OFF-state. A.7.1.3 The mute function shall only be initiated by the correct sequence and/or timing of the muting signals. Should conflicting muting signals occur, the ESPE shall not allow a muted condition to occur. A.7.1.4 There shall be at least two independent hard-wired muting signal sources to stop the function. The muting function shall stop when the first of these muting signals changes state. The deactivation of the muting function shall not rely only on the clearance of the ESPE. A.7.1.5 The muting signals should be continuously present during muting. When the signals are not continuously present, an incorrect sequence and/or the expiration of a pre-set time limit shall cause either a lock-out condition or a restart interlock. A.7.4 Indication: A mute status signal or indicator shall be provided (in some applications, an indication signal of muting is necessary)
IEC 62046/ Ed. 1:2005	5.5.1: ... an indicator to show when the muting function is active can be necessary. The muting function shall be initiated and terminated automatically ... Incorrect signals, sequence or timing of the muting sensors or signals shall not allow a mute condition. It shall not be possible to initiate the muting function when: ● the protective equipment OSSDs are in the OFF-state; ● the protective equipment is in the lock-out condition; ● initiation of the muting function by two or more independent muting sensors such that a single fault cannot cause a muted condition; ● termination of the muting function by two or more independent muting sensors such that deactivation of one sensor will terminate the muting function; ● use of timing and sequence control of the muting sensors to ensure correct muting operation. 5.5.3: The following measures shall be considered ... ● limiting muting to a fixed time that is only sufficient for the material to pass through the detection zone. When this time is exceeded, the muting function should be canceled and all hazardous movements stopped. Annex F.7 Two sensors - Crossed beams (refer also to Fig. F.7.2 and F.7.3 in the standard) The muting function should only be initiated when the two beams are activated within a time limit of 4 sec. The muting function should be terminated as soon as one of the two beams of the muting sensors is no longer activated. A monitored timer that limits the muting function to the minimum practicable time is required. Annex F.5: Methods to avoid manipulation of the muting function: ... use a muting enable command generated by the control system of the machine that will only enable the muting function when needed by the machine cycle.
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



Muting is the intended suppression of the safety function. In this FB, parallel muting with two muting sensors is specified.

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be push buttons, proximity switches, photoelectric barriers, limit switches, etc. which do not have to be fail-safe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, parallel muting with two muting sensors was used; an explanation is provided below. The positioning of the sensors should be as described in Annex F.7 of IEC 62046, 2005 *“Example for SF_MutingPar_2Sensor with two reflecting light barriers” on page 275*. The FB can be used in both directions, forward and backward. However, the actual direction cannot be identified. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation.

The FB input parameters include the signals of the two muting sensors (S_MutingSwitch11 and S_MutingSwitch12), the OSSD signal from the AOPD device (S_AOPD_In) as well as two parameterizable times (DiscTimeEntry and MaxMutingTime).

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 57: FB name: SF_MutingPar_2Sensor

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_AOPD_In	BOOL	FALSE	Variable. OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
S_MutingSwitch11	BOOL	FALSE	Variable. Status of Muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11.
S_MutingSwitch12	BOOL	FALSE	Variable. Status of Muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12.

Name	Data type	Initial value	Description, parameter values
S_MutingLamp	BOOL	FALSE	Variable or constant. Indicates operation of the muting lamp. FALSE: Muting lamp failure. TRUE: No muting lamp failure.
MutingEnable	BOOL	FALSE	Variable or constant. Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of Muting function enabled
S_StartReset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
Reset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
DiscTimeEntry	TIME	T#0s	Constant 0..4 s; Max. discrepancy time for S_MutingSwitch11 and S_MutingSwitch12 entering muting gate
MaxMutingTime	TIME	T#0s	Constant 0..10 min; Maximum time for complete muting sequence, timer started when first muting sensor is actuated.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
S_AOPD_Out	BOOL	FALSE	Safety related output, indicates status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
S_MutingActive	BOOL	FALSE	Indicates status of Muting process. FALSE: Muting not active. TRUE: Muting active.
Error	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
DiagCode	WORD	16#0000	*Table 17 "General output parameters" on page 200*

Note: Line control of muting sensor signals must be active in the safety loop.

Example for SF_MutingPar_2 Sensor with two reflecting light barriers

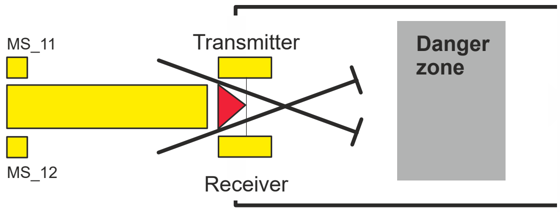


Fig. 110: Example for SF_MutingPar_2Sensor

If reflection light barriers are used as muting sensors, they are generally arranged diagonally. In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only S_MutingSwitch11 (MS_11) and S_MutingSwitch12 (MS_12) are allocated.

Muting conditions

Muting condition 1 (to state 8011) (MS_11 is the first actuated entry switch). Start timer DiscTimeEntry and MaxMutingTime:

MutingEnable AND R_TRIG at MS_11 AND NOT MS_12

Muting condition 2 (to state 8311) (MS_12 is the first actuated entry switch). Start timer DiscTimeEntry and MaxMutingTime:

MutingEnable AND NOT MS_11 AND R_TRIG at MS_12

Muting condition 3 (from state 8011 to state 8012) (MS_12 is the second actuated entry switch):

Stop timer DiscTimeEntry:

MutingEnable AND MS_11 AND R_TRIG at MS_12

Muting condition 4 (from state 8311 to state 8012) (MS_11 is the second actuated entry switch):

Stop timer DiscTimeEntry:

MutingEnable AND R_TRIG at MS_11 AND MS_12

Muting condition 5 (from state 8000 to state 8012) (both switches actuated in same cycle): Start Timer MaxMutingTime:

MutingEnable AND R_TRIG at MS_11 AND R_TRIG at MS_12

Muting condition 6 (from state 8012 to state 8000) (both switches released in same cycle or MS_11 and MS_12 released consecutively). Stop timer MaxMutingTime: NOT MS_11 OR NOT MS_12

Wrong muting sequences

State 8000 - (R_TRIG at MS_11 AND MS_12 AND NOT R_TRIG at MS_12) OR

(R_TRIG at MS_12 AND MS_11 AND NOT R_TRIG at MS_11) OR

((MS_11 AND NOT R_TRIG at MS_11) AND (MS_12 AND NOT R_TRIG at MS_12)) OR

(NOT MutingEnable AND R_TRIG at MS_11) OR

(NOT MutingEnable AND R_TRIG at MS_12)

State 8011 - NOT MutingEnable OR NOT MS_11

State 8311 - NOT MutingEnable OR NOT MS_12

State 8012 - all possible transitions allowed

Typical timing diagram

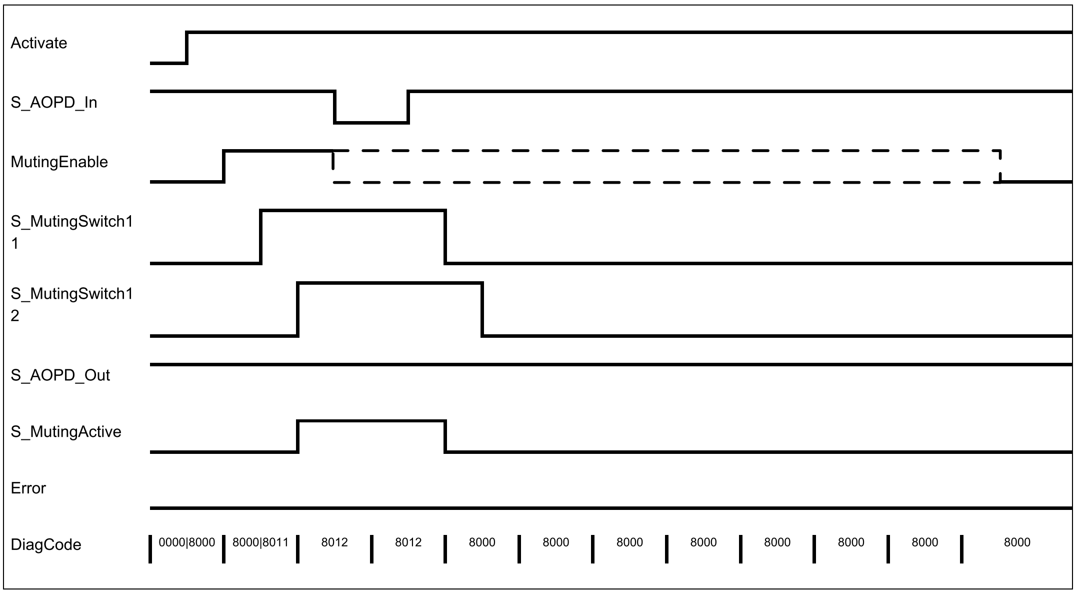


Fig. 111: Timing diagram for SF_MutingPar_2Sensor (S_StartReset = TRUE, Reset = FALSE, S_MutingLamp = TRUE)

The FB detects the following error conditions:

- DiscTimeEntry has been set to value less than T#0s or greater than T#4s.
- MaxMutingTime has been set to a value less than T#0s or greater than T#10min.
- The discrepancy time for the S_MutingSwitch11/S_MutingSwitch12 sensor pair has been exceeded.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors S_MutingSwitch11, S_MutingSwitch12 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable.
- Static muting sensor signals.
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static Reset condition is detected in state 8001 and 8003.

Error behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.

A restart is inhibited until the error conditions are cleared and the safe state is acknowledged with Reset by the operator.

Function block-specific error and status codes

Table 58: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error 1	Static Reset condition detected after FB activation in state 8001. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

DiagCode	State name	State description and output setting
C003	Error Muting Lamp	Error detected in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
CYx4	Error Muting sequence	Error detected in muting sequence state 8000, 8011, 8311. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state 8011 C2x4 = Error occurred in state 8311 CFx4 = Muting enable missing x = Status of the sensors when error occurred (4 bits: LSB = MS_11; next to LSB = MS_12).
C005	Parameter Error	DiscTimeEntry or MaxMutingTime value out of range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C007	Error timer Entry	Timing error: Discrepancy time for switching S_MutingSwitch11 and S_MutingSwitch12 from FALSE to TRUE > DiscTimeEntry. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

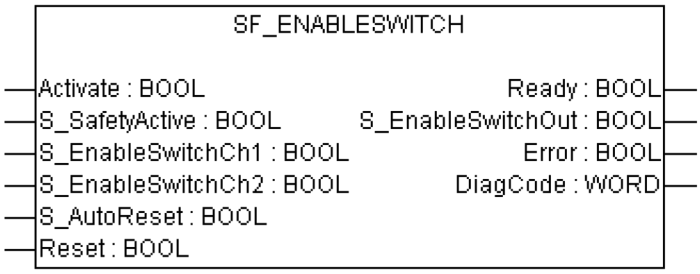
Table 59: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety demand from AOPD. If timers from subsequent muting are still running, they are stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block was activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety demand detected by AOPD, muting not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Start 1	Muting sequence is in starting phase after rising trigger of S_MutingSwitch11. Monitoring of DiscTimeEntry is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8311	Muting Start 2	Muting sequence is in starting phase after rising trigger of S_MutingSwitch12. Monitoring of DiscTimeEntry is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8012	Muting Active	Muting sequence is active either: - After rising trigger of the second S_MutingSwitch 12 or 11 has been detected. - When both S_MutingSwitch 11 and 12 have been actuated in the same cycle. Monitoring of DiscTimeEntry is stopped. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE

4.6.4.15 SF_EnableSwitch

Standards	Requirements
IEC 60204-1, Ed. 5.0:2003	9.2.6.3: Enabling control (refer also to 10.9 below) is a manually activated control function interlock that: ● when activated allows a machine operation to be initiated by a separate start control, and ● when deactivated - initiates a stop function, and - prevents initiation of machine operation. Enabling control shall be so arranged as to minimize the possibility of defeating, for example, by requiring the deactivation of the enabling control device before machine operation may be reinitiated. It should not be possible to defeat the enabling function by simple means. 10.9: When an enabling control device is provided as a part of a system, it shall signal the enabling control to allow operation when actuated in one position only. In any other position, operation shall be stopped or prevented. Enabling control devices shall be selected that have the following features: ... ● for a three-position type: – position 1: off-function of the switch (actuator is not operated); – position 2: enabling function (actuator is operated in its mid position); – position 3: off-function (actuator is operated past its mid position); ● when returning from position 3 to position 2, the enabling function is not activated.
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



The SF_EnableSwitch FB evaluates the signals of an enable switch with three positions.

The SF_EnableSwitch FB supports the suspension of safeguarding (EN 60204 Section 9.2.4) using enable switches (EN 60204 Section 9.2.5.8), if the relevant operating mode is selected and active. The relevant operating mode (limitation of the speed or the power of motion, limitation of the range of motion) must be selected outside the SF_EnableSwitch FB.

The SF_EnableSwitch FB evaluates the signals of an enable switch with three positions (EN 60204 Section 9.2.5.8).

The S_EnableSwitchCh1 and S_EnableSwitchCh2 input parameters process the following signal levels of contacts E1 to E4:

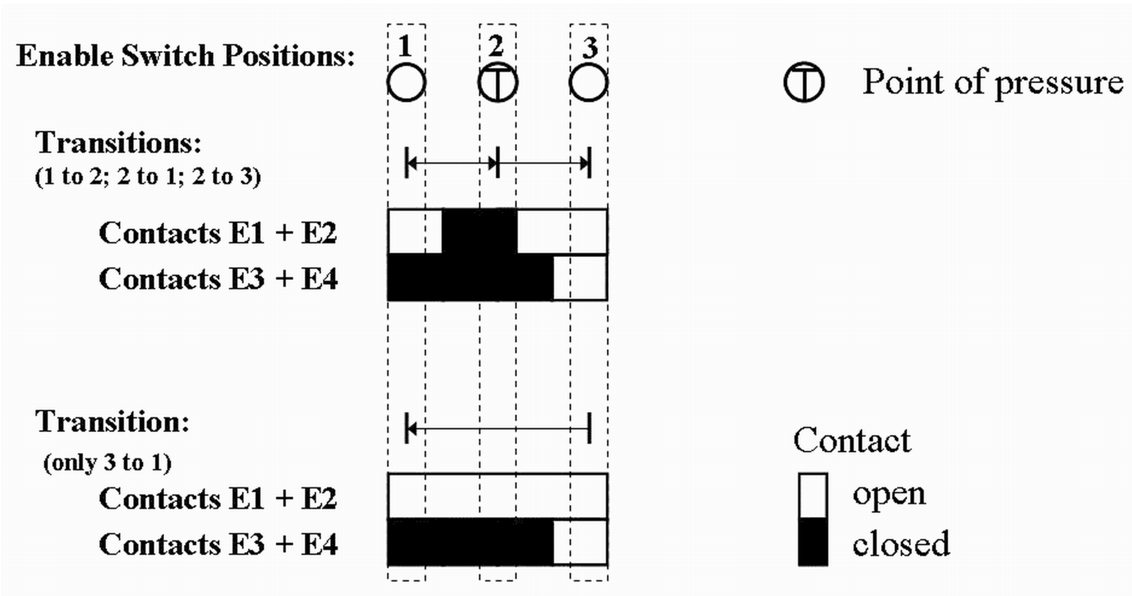


Fig. 112: Switch positions

The signal from E1+E2 must be connected to the S_EnableSwitchCh1 parameter. The signal from E3+E4 must be connected to the S_EnableSwitchCh2 parameter. The position of the enable switch is detected in the FB using this signal sequence.

The transition from position 2 to 3 can be different from shown here.

The switching direction (position 1 => position 2/position 3 => position 2) can be detected in the FB using the defined signal sequence of the enable switch contacts. The suspension of safeguarding can only be enabled by the FB after a move from position 1 to position 2. Other switching directions or positions may not be used to enable the suspension of safeguarding. This measure meets the requirements of EN 60204 Section 9.2.5.8.

In order to meet the requirements of EN 60204 Section 9.2.4, the user shall use a suitable switching device. In addition, the user must ensure that the relevant operating mode (EN 60204 Section 9.2.3) is selected in the application (automatic operation must be disabled in this operating mode using appropriate measures).

The operating mode is usually specified using an operating mode selection switch in conjunction with SF_ModeSelector FB and SF_SafeRequest or SF_SafelyLimitedSpeed FB.

The SF_EnableSwitch FB processes the confirmation of the "safe mode" state via the "S_SafetyActive" parameter. On implementation in an application of the safe mode without confirmation, a static TRUE signal is connected to the "S_SafetyActive" parameter.

The S_AutoReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 60: FB name: SF_EnableSwitch

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_SafetyActive	BOOL	FALSE	Variable or constant. Confirmation of the safe mode (limitation of the speed or the power of motion, limitation of the range of motion). FALSE: Safe mode is not active. TRUE: Safe mode is active.
S_Enable-SwitchCh1	BOOL	FALSE	Variable. Signal of contacts E1 and E2 of the connected enable switch. FALSE: Connected switches are open. TRUE: Connected switches are closed.
S_Enable-SwitchCh2	BOOL	FALSE	Variable. Signal of contacts E3 and E4 of the connected enable switch. FALSE: Connected switches are open. TRUE: Connected switches are closed.
S_AutoReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_EnableSwitchOut	BOOL	FALSE	Safety related output: Indicates suspension of guard. FALSE: Disable suspension of safeguarding. TRUE: Enable suspension of safeguarding.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Typical timing diagrams

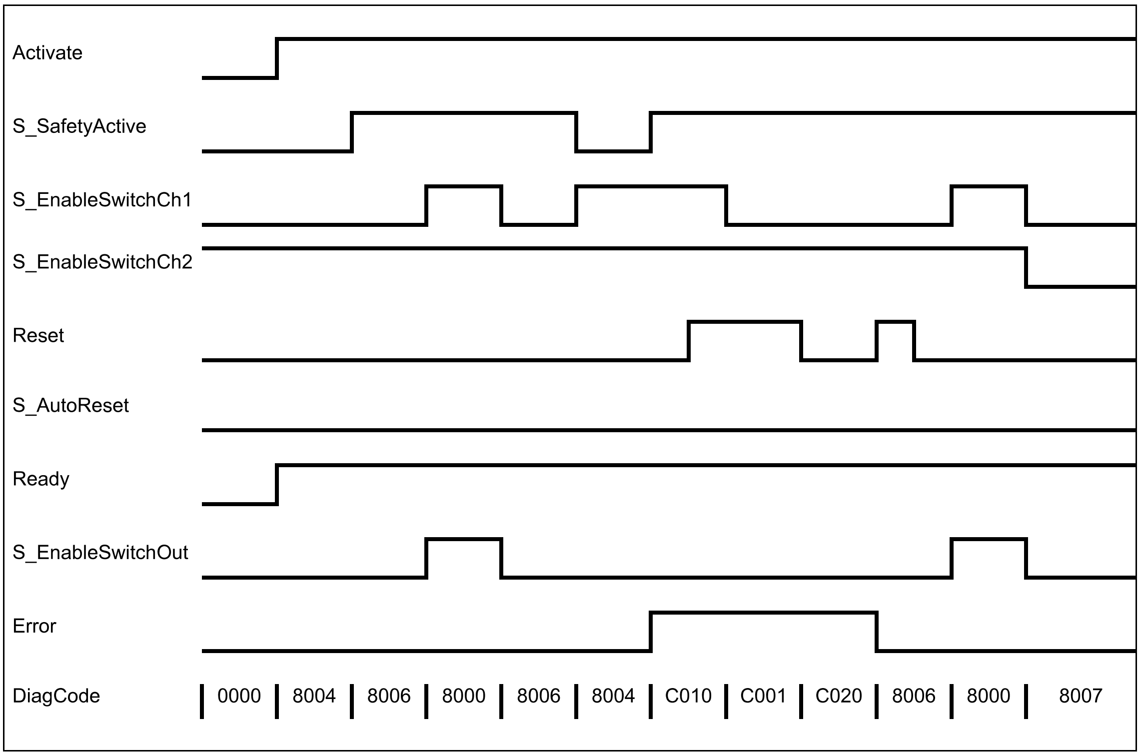


Fig. 113: Timing diagram for SF_EnableSwitch: S_AutoReset = FALSE

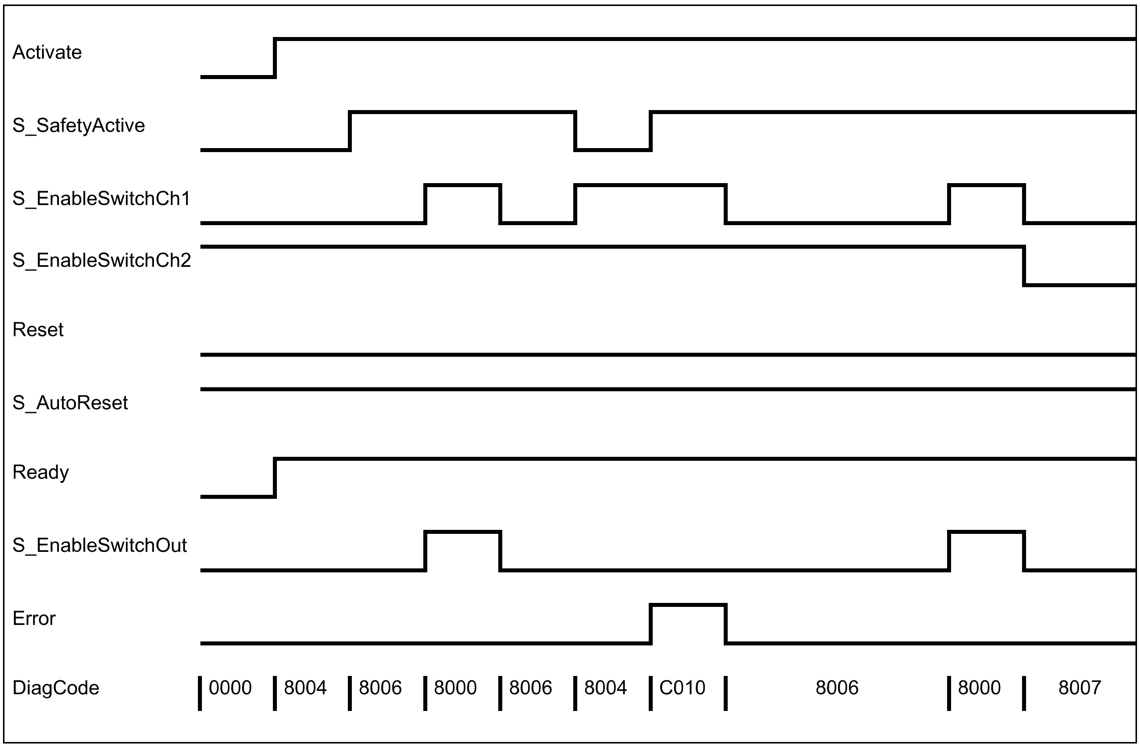


Fig. 114: Timing diagram for SF_EnableSwitch: S_AutoReset = TRUE

The following conditions force a transition to the error state:

- Invalid static Reset signal in the process.
- Invalid switch positions.

Error behavior

In the event of an error, the S_EnableSwitchOut safe output is set to FALSE and remains in this safe state.

Different from other FBs, a reset error state can be left by the condition Reset = FALSE or, additionally, when the signal S_SafetyActive is FALSE.

Once the error has been removed, the enable switch must be in the initial position specified in the process before the S_EnableSwitchOut output can be set to TRUE using the enable switch. If S_AutoReset = FALSE, a rising trigger is required at Reset.

Function block-specific error and status codes

Table 61: FB-specific error codes

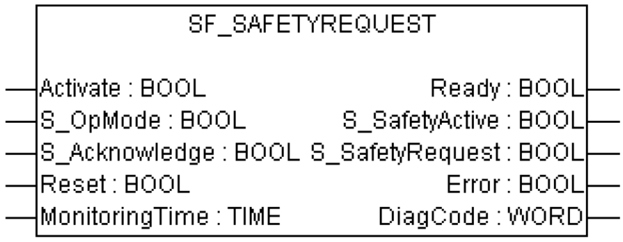
DiagCode	State name	State description and output setting
C001	Reset Error 1	Static Reset signal detected in state C020. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C002	Reset Error 2	Static Reset signal detected in state C040. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C010	Operation Error 1	Enable switch not in position 1 during activation of S_SafetyActive. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C020	Operation Error 2	Enable switch in position 1 after C010. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C030	Operation Error 3	Enable switch in position 2 after position 3. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C040	Operation Error 4	Enable switch not in position 2 after C030. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE

Table 62: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EnableSwitchOut = FALSE Error = FALSE
8004	Basic Operation Mode	Safe operation mode is not active. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8005	Safe Operation Mode	Safe operation mode is active. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8006	Position 1	Safe operation mode is active and the enable switch is in position 1. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8007	Position 3	Safe operation mode is active and the enable switch is in position 3. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8000	Position 2	Safe operation mode is active and the enable switch is in position 2. Ready = TRUE S_EnableSwitchOut = TRUE Error = FALSE

4.6.4.16 SF_SafetyRequest

Standards	Requirements
IEC 60204-1, Ed. 5.0:2003	9.2.4 Suspension of safety functions and/or protective measures Where it is necessary to suspend safety functions and/or protective measures (for example, for setting or maintenance purposes), protection shall be ensured by: ● disabling all other operating (control) modes; and ● other relevant means (refer to 4.11.9 of ISO 12100-2:2003), that can include, for example, one or more of the following: – limitation of the speed or the power of motion – limitation of the range of motion
EN 954-1:1996	5.4 Manual reset
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart



The function block represents the interface between the user program and system environment.

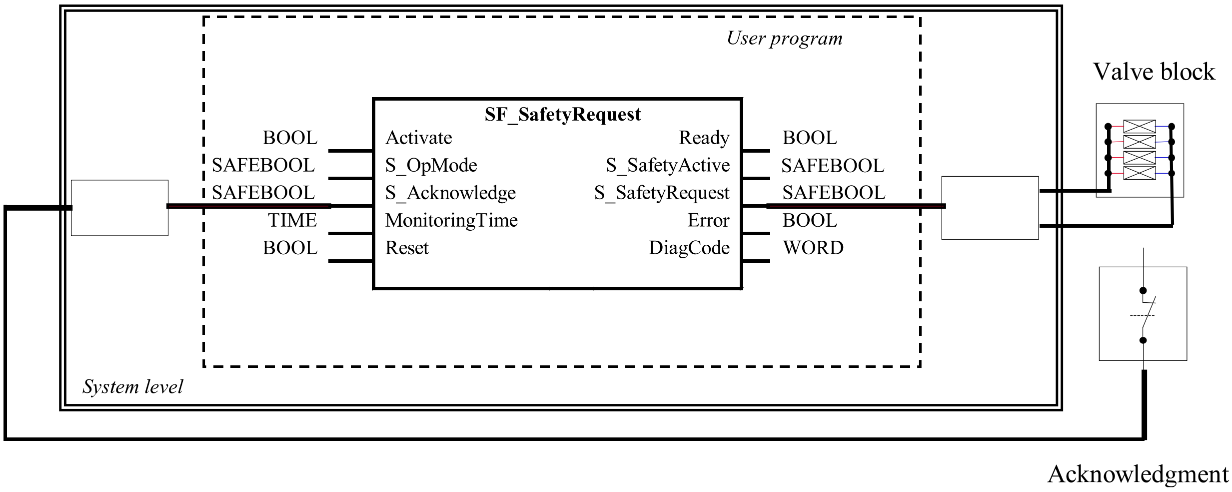


Fig. 115: Example of SF_SafetyRequest

This function block provides the interface to a generic actuator, e.g. a safety drive or safety valve, to place the actuator in a safe state.

This FB provides the interface between the safety-related system and a generic actuator. This means that the safety-related functions of the actuator are available within the application program. However, there are only two binary signals to control the safe state of the generic actuator, i.e., one for requesting and one for receiving the confirmation.

The safety function will be provided by the actuator itself. Therefore the FB only initiates the request, monitors it, and sets the output when the actuator acknowledges the safe state. This will be indicated with the S_SafetyActive output.

This FB does not define any generic actuator-specific parameters. They should have been specified in the generic actuator itself. It switches the generic actuator from the operation mode to a safe state.

Table 63: FB name: SF_SafetyRequest

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_OpMode	BOOL	FALSE	Variable. Requested mode of a generic safe actuator. FALSE: Safe mode is requested. TRUE: Operation mode is requested.

Name	Data type	Initial value	Description, parameter values
S_Acknowledge	BOOL	FALSE	Variable. Confirmation of the generic actuator, if actuator is in the Safe state. FALSE: Operation mode (non-safe). TRUE: Safe mode.
Reset	BOOL	FALSE	*Table 16 "General input parameters" on page 199*
MonitoringTime	TIME	T#0s	Constant. Monitoring of the response time between the safety function request (S_OpMode set to FALSE) and the actuator acknowledgment (S_Acknowledge switches to TRUE).
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
S_SafetyActive	BOOL	FALSE	Confirmation of the safe state. FALSE: Non-safe state. TRUE: Safe state.
S_SafetyRequest	BOOL	FALSE	Request to place the actuator in a safe state. FALSE: Safe state is requested. TRUE: Non-safe state.
Error	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
DiagCode	WORD	16#0000	*Table 17 "General output parameters" on page 200*

Typical timing diagram

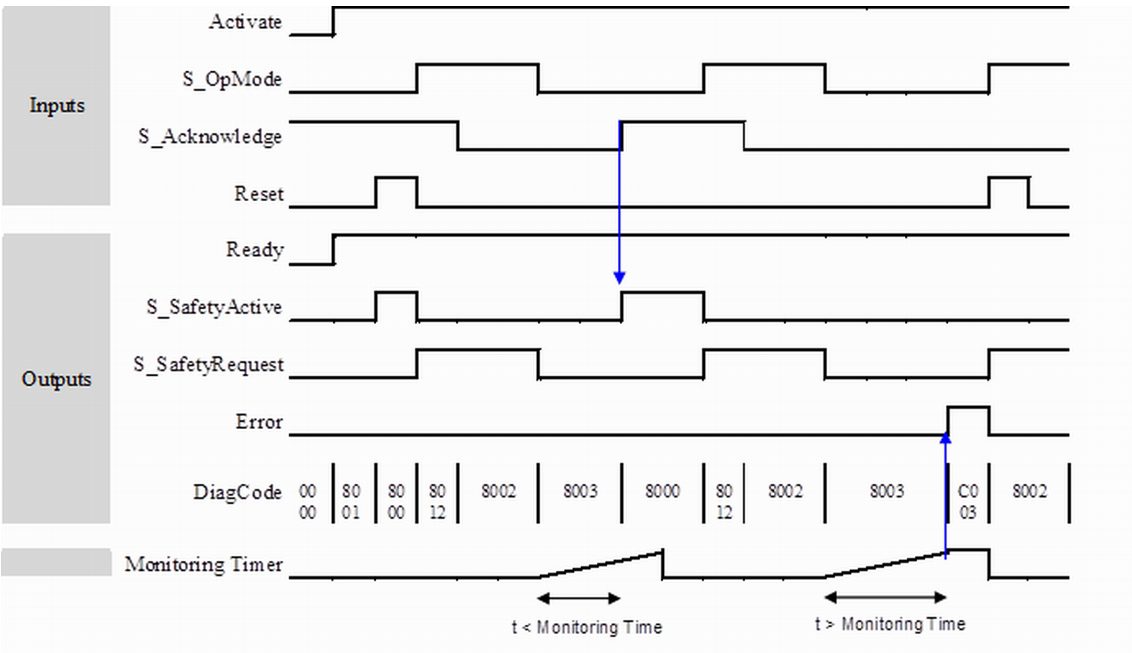


Fig. 116: Timing diagram for SF_SafetyRequest

The FB detects whether the actuator does not enter the safe state within the monitoring time.

The FB detects whether the acknowledge signal is lost while the request is still active.

The FB detects a static Reset signal.

External FB errors: There are no external errors, since there is no error bits/information provided by the generic actuator.

Error behavior

In the event of an error, the S_SafetyActive output is set to FALSE.

An error must be acknowledged by a rising trigger at the Reset input. To continue the function block after this reset, the S_OpMode request must be set to TRUE.

Function block-specific error and status codes

Table 64: FB-specific error codes

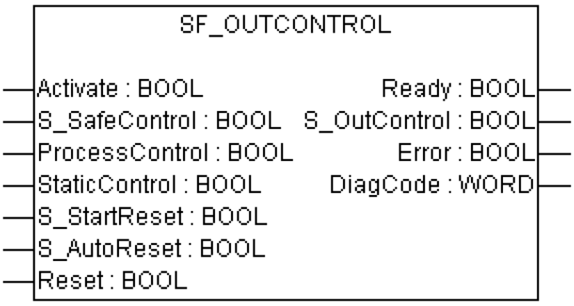
DiagCode	State name	State description and output setting
C002	Acknowl-edge Lost	Acknowledgment lost while in the safe state. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C003	Monitoring-Time Elapsed	S_OpMode request could not be completed within the monitoring time. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C004	Reset Error 2	Static Reset detected in state C002 (Acknowledge Lost). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C005	Reset Error 3	Static Reset detected in state C003 (MonitoringTime elapsed). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE

Table 65: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8000	Safe Mode	Actuator is in a safe mode. Ready = TRUE S_SafetyActive = TRUE S_SafetyRequest = FALSE Error = FALSE
8001	Init	State after Activate is set to TRUE or after a rising trigger at Reset. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8002	Operation Mode	Operation mode without Acknowledge of safe mode Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE Error = FALSE
8012	Wait for Confirmation OpMode	Operation mode with Acknowledge of safe mode Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE Error = FALSE
8003	Wait for Confirmation	Waiting for confirmation from the drive (system interface). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8005	Wait for OpMode	Error was cleared. However S_OpMode must be set to TRUE before the FB can be initialized. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE

4.6.4.17 SF_OutControl

Standards	Requirements
IEC 60204-1, Ed. 5.0:2003	9.2.2: Stop functions: Stop function categories; Category 0 - stopping by immediate removal of power to the machine actuators (i.e. an uncontrolled stop ...) 9.2.5.2: Start: The start of an operation shall be possible only when all of the relevant safety functions and/or protective measures are in place and are operational except for conditions as described in 9.2.4. Suitable interlocks shall be provided to secure correct sequential starting.
EN 954-1:1996	5.2: Stop function; stop initiated by protective devices shall put the machine in a safe state ... and shall have priority over a stop for operational reasons. 5.5: Start and restart; automatic restart only if a hazardous situation cannot exist. 5.11: Fluctuations in energy levels; in case of loss of energy supply, provide or initiate outputs to maintain a safe state.
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart
EN 954-1:1996	5.4 Manual reset



Control of a safety output with a signal from the functional application and a safety signal with optional startup inhibits.

The SF_OutControl FB is an output driver for a safety output.

The safety output is controlled via S_OutControl using a signal from the functional application (ProcessControl to control the process) and a signal from the safety application (S_SafeControl to control the safety function).

Optional conditions for process control (ProcessControl):

- An additional function start (ProcessControl FALSE => TRUE) is required following block activation or feedback of the safe signal (S_SafeControl). A static TRUE signal at ProcessControl does not set S_OutControl to TRUE.
- An additional function start (ProcessControl FALSE => TRUE) is not required following block activation or feedback of the safe signal (S_SafeControl). A static TRUE signal at ProcessControl sets S_OutControl to TRUE if the other conditions have been met.

Optional startup inhibits:

- Startup inhibit after function block activation.
- Startup inhibit after interruption of the protective device.

The StaticControl, S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 66: FB name: SF_OutControl

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	➪ *Table 16 “General input parameters” on page 199*

Name	Data type	Initial value	Description, parameter values
S_SafeControl	BOOL	FALSE	Variable. Control signal of the preceding safety FB. Typical function block signals from the library (e.g., SF_EStop, SF_GuardMonitoring, SF_TwoHandControlTypeII, and/or others). FALSE: The preceding safety FBs are in safe state. TRUE: The preceding safety FBs enable safety control.
ProcessControl	BOOL	FALSE	Variable or constant. Control signal from the functional application. FALSE: Request to set S_OutControl to FALSE. TRUE: Request to set S_OutControl to TRUE.
StaticControl	BOOL	FALSE	Constant. Optional conditions for process control. FALSE: Dynamic change at ProcessControl (FALSE => TRUE) required after block activation or triggered safety function. Additional function start required. TRUE: No dynamic change at ProcessControl (FALSE => TRUE) required after block activation or triggered safety function.
S_StartReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_AutoReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_OutControl	BOOL	FALSE	Controls connected actuators. FALSE: Disable connected actuators. TRUE: Enable connected actuators.
Error	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
DiagCode	WORD	16#0000	*Table 17 “General output parameters” on page 200*

Typical timing diagrams

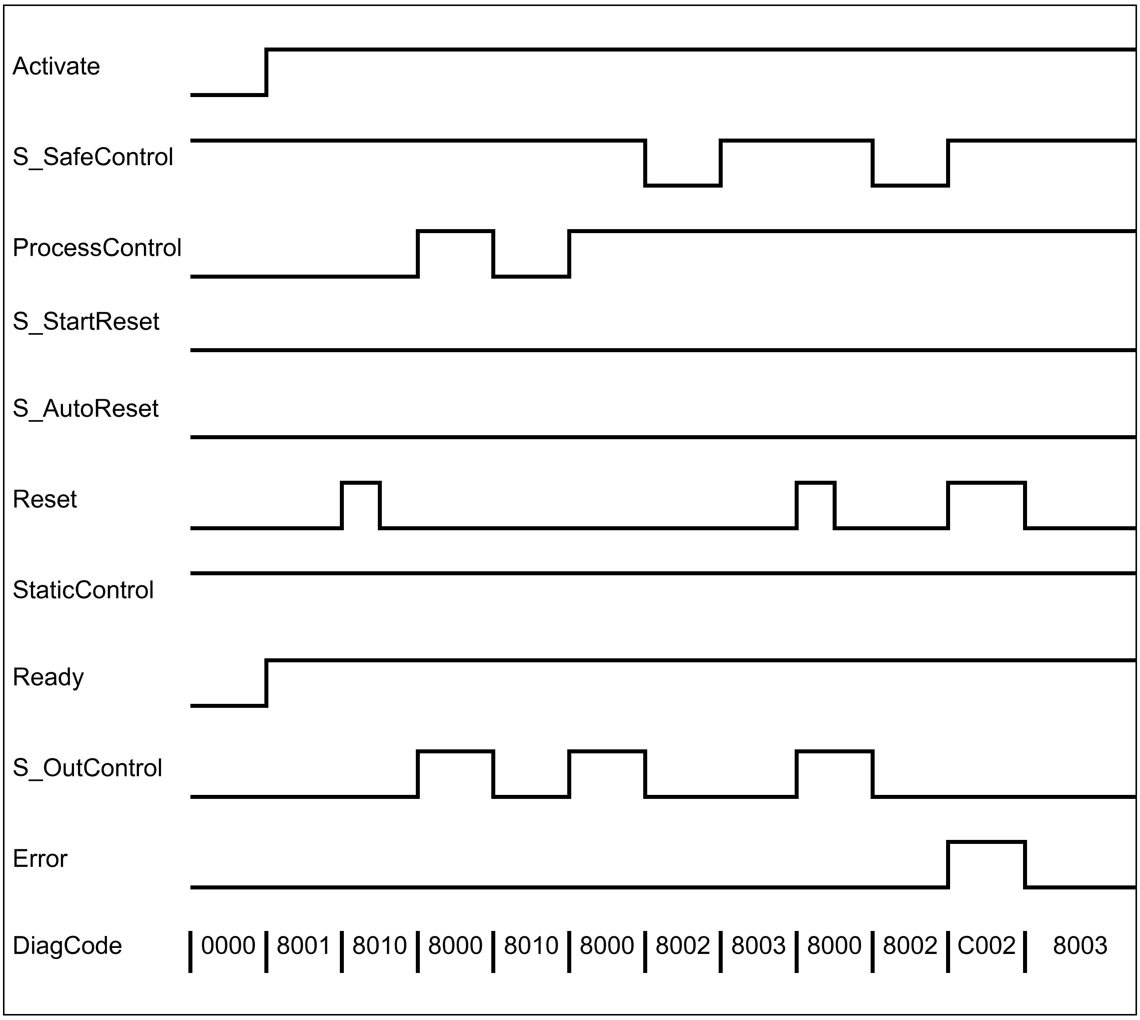


Fig. 117: Timing diagram for SF_OutControl: S_StartReset = FALSE

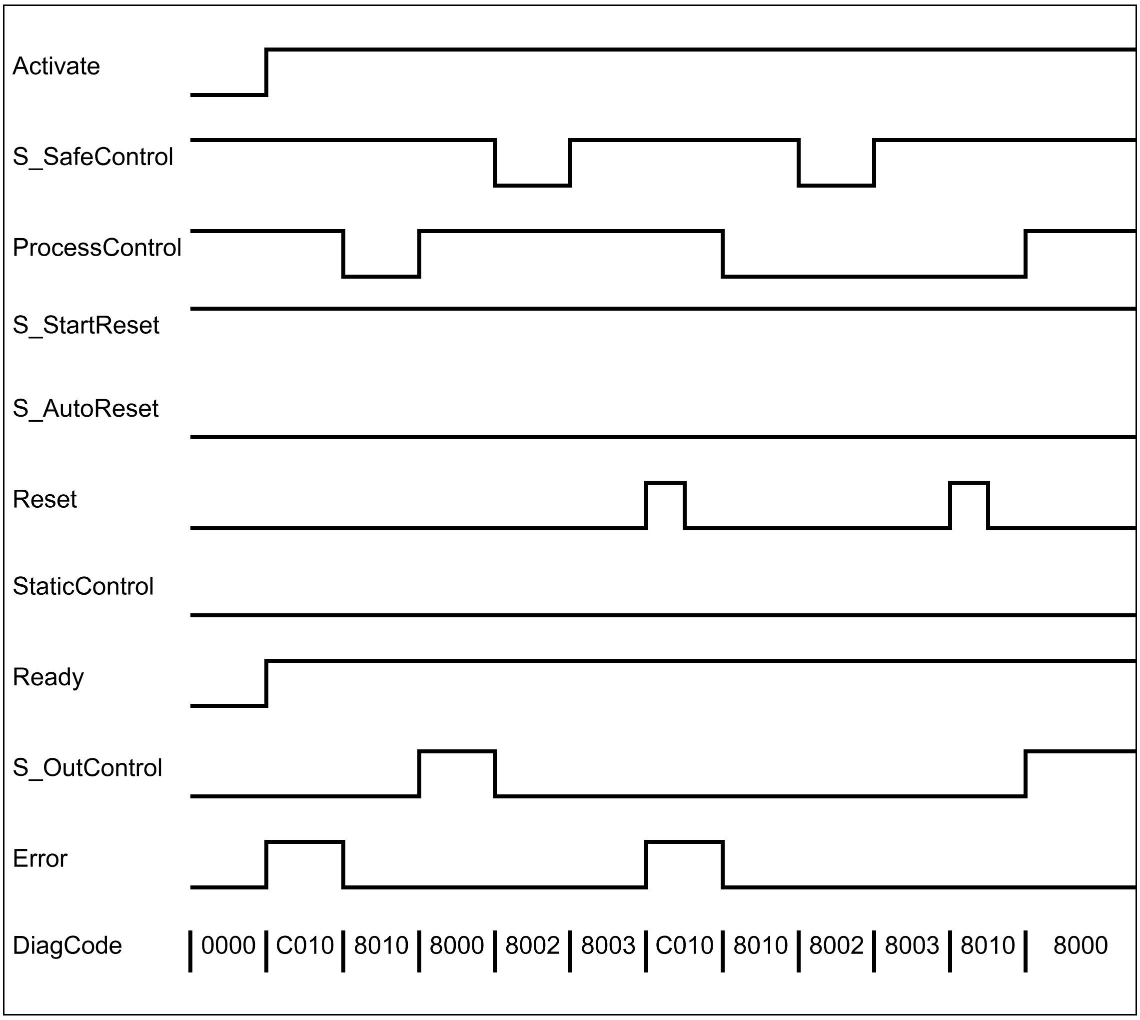


Fig. 118: Timing diagram for SF_OutControl: S_StartReset = TRUE

The following conditions force a transition to the Error state:

- Invalid static Reset signal in the process.
- Invalid static ProcessControl signal.
- ProcessControl and Reset are incorrectly interconnected due to programming error.

Error behavior

In the event of an error, the S_OutControl output is set to FALSE and remains in this safe state.

To leave the Reset, Init or Lock error states, the Reset input must be set to FALSE. To leave the control error state, the ProcessControl input must be set to FALSE.

After transition of S_SafeControl to TRUE, the optional startup inhibit can be reset by a rising edge at the Reset input.

After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

Function block-specific error and status codes

Table 67: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error 1	Static Reset signal in state 8001. Ready = TRUE S_OutControl = FALSE Error = TRUE
C002	Reset Error 2	Static Reset signal in state 8003. Ready = TRUE S_OutControl = FALSE Error = TRUE
C010	Control Error	Static signal at ProcessControl in state 8010. Ready = TRUE S_OutControl = FALSE Error = TRUE
C111	Init Error	Simultaneous rising trigger at Reset and ProcessControl in state 8001. Ready = TRUE S_OutControl = FALSE Error = TRUE
C211	Lock Error	Simultaneous rising trigger at Reset and ProcessControl in state 8003. Ready = TRUE S_OutControl = FALSE Error = TRUE

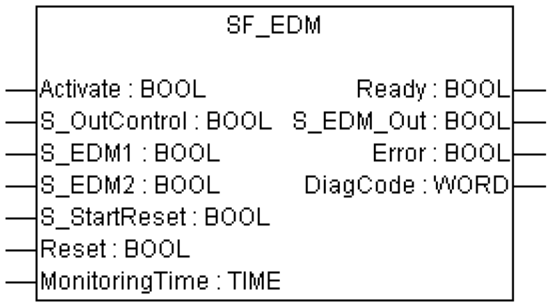
Table 68: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_OutControl = FALSE Error = FALSE
8001	Init	Block activation startup inhibit is active. Reset required. Ready = TRUE S_OutControl = FALSE Error = FALSE
8002	Safe	Triggered safety function. Ready = TRUE S_OutControl = FALSE Error = FALSE

DiagCode	State name	State description and output setting
8003	Lock	Safety function startup inhibit is active. Reset required. Ready = TRUE S_OutControl = FALSE Error = FALSE
8010	Output Dis-able	Process control is not active. Ready = TRUE S_OutControl = FALSE Error = FALSE
8000	Output Enable	Process control is active and safety is enabled. Ready = TRUE S_OutControl = TRUE Error = FALSE

4.6.4.18 SF_EDM

Standards	Requirements
IEC 60204-1, Ed. 5.0:2003	Section 9.2.2: Stop function categories; Category 0
EN 954-1:1996	5.2: Stop function; stop initiated by protective devices shall put the machine in a safe state 6.2: Specification of categories: Fault detection (of the actuator, e.g. open circuits)
ISO 12100-2:2003	4.11.4: Restart following power failure/spontaneous restart
EN 954-1:1996	5.4 Manual reset



External device monitoring (EDM): The FB controls a safety output and monitors controlled actuators, e.g. subsequent contactors.

The SF_EDM FB controls a safety output and monitors controlled actuators.

This function block monitors the initial state of the actuators via the feedback signals (S_EDM1 and S_EDM2) before the actuators are enabled by the FB.

The function block monitors the switching state of the actuators (MonitoringTime) after the actuators have been enabled by the FB.

Two single feedback signals must be used for an exact diagnosis of the connected actuators. A common feedback signal from the two connected actuators must be used for a restricted yet simple diagnostic function of the connected actuators. When doing so, the user must connect this common signal to both parameter S_EDM1 and parameter S_EDM2. S_EDM1 and S_EDM2 are then controlled by the same signal.

The switching devices used in the safety function should be selected from the category specified in the risk analysis (EN 954-1).

Optional startup inhibits:

- Startup inhibit in the event of block activation.

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Table 69: FB name: SF_EDM

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Activate	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
S_OutControl	BOOL	FALSE	Variable. Control signal of the preceding safety FBs. Typical function block signals from the library (e.g., SF_OutControl, SF_TwoHandControlTypeII, and/or others). FALSE: Disable safety output (S_EDM_Out). TRUE: Enable safety output (S_EDM_Out).
S_EDM1	BOOL	FALSE	Variable. Feedback signal of the first connected actuator. FALSE: Switching state of the first connected actuator. TRUE: Initial state of the first connected actuator.
S_EDM2	BOOL	FALSE	Variable. Feedback signal of the second connected actuator. If using only one signal in the application, the user must use a graphic connection to jumper the S_EDM1 and S_EDM2 parameters. S_EDM1 and S_EDM2 are then controlled by the same signal. FALSE: Switching state of the second connected actuator. TRUE: Initial state of the second connected actuator.
S_StartReset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
Reset	BOOL	FALSE	*Table 16 “General input parameters” on page 199*
MonitoringTime	TIME	#0ms	Constant. Max. response time of the connected and monitored actuators.
VAR_OUTPUT			
Ready	BOOL	FALSE	*Table 17 “General output parameters” on page 200*
S_EDM_Out	BOOL	FALSE	Controls the actuator. The result is monitored by the feedback signal S_EDMx. FALSE: Disable connected actuators. TRUE: Enable connected actuators.

Name	Data type	Initial value	Description, parameter values
Error	BOOL	FALSE	*Table 17 "General output parameters" on page 200*
DiagCode	WORD	16#0000	*Table 17 "General output parameters" on page 200*

Typical timing diagrams

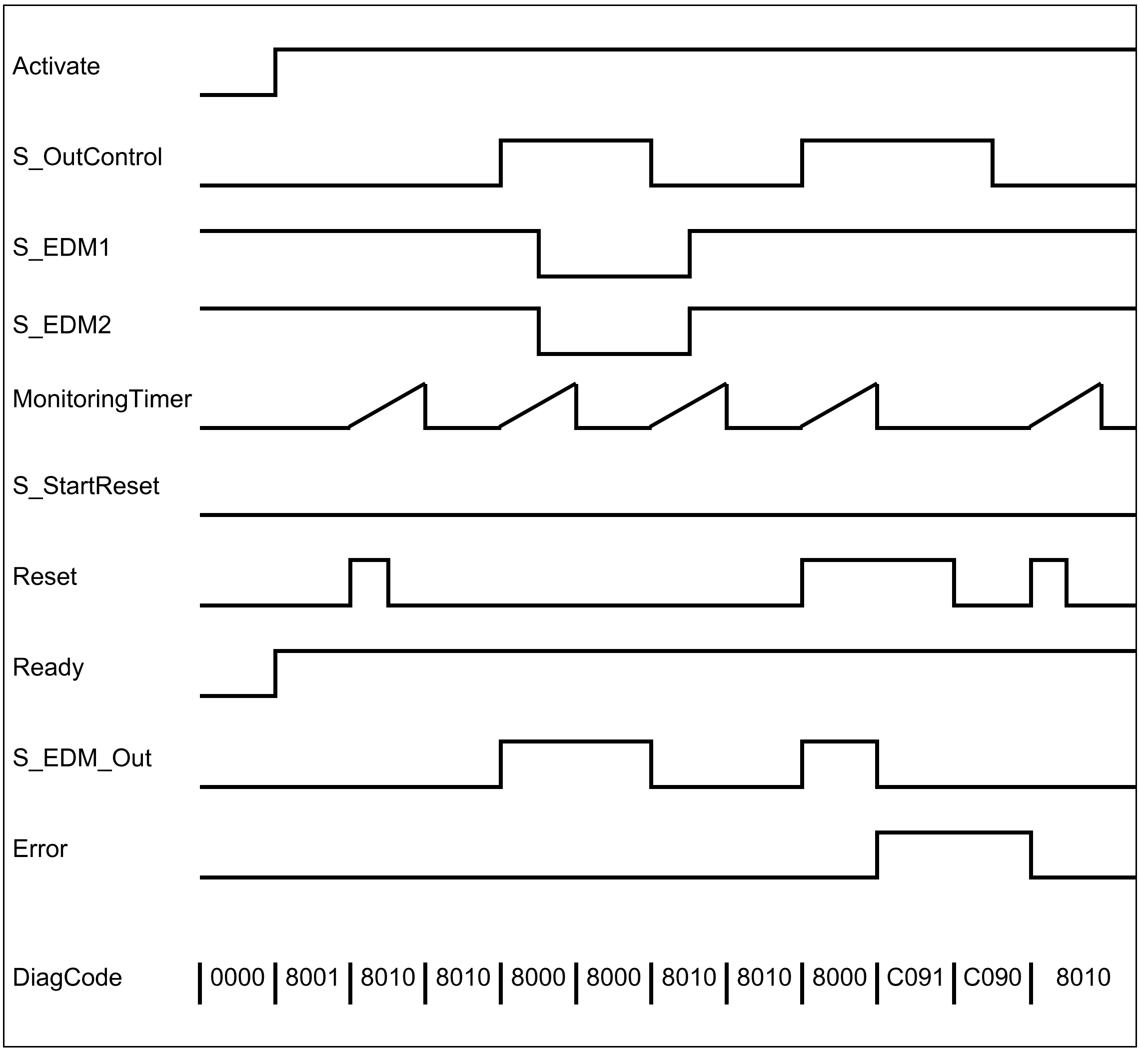


Fig. 119: Timing diagrams for SF_EDM: S_StartReset = FALSE

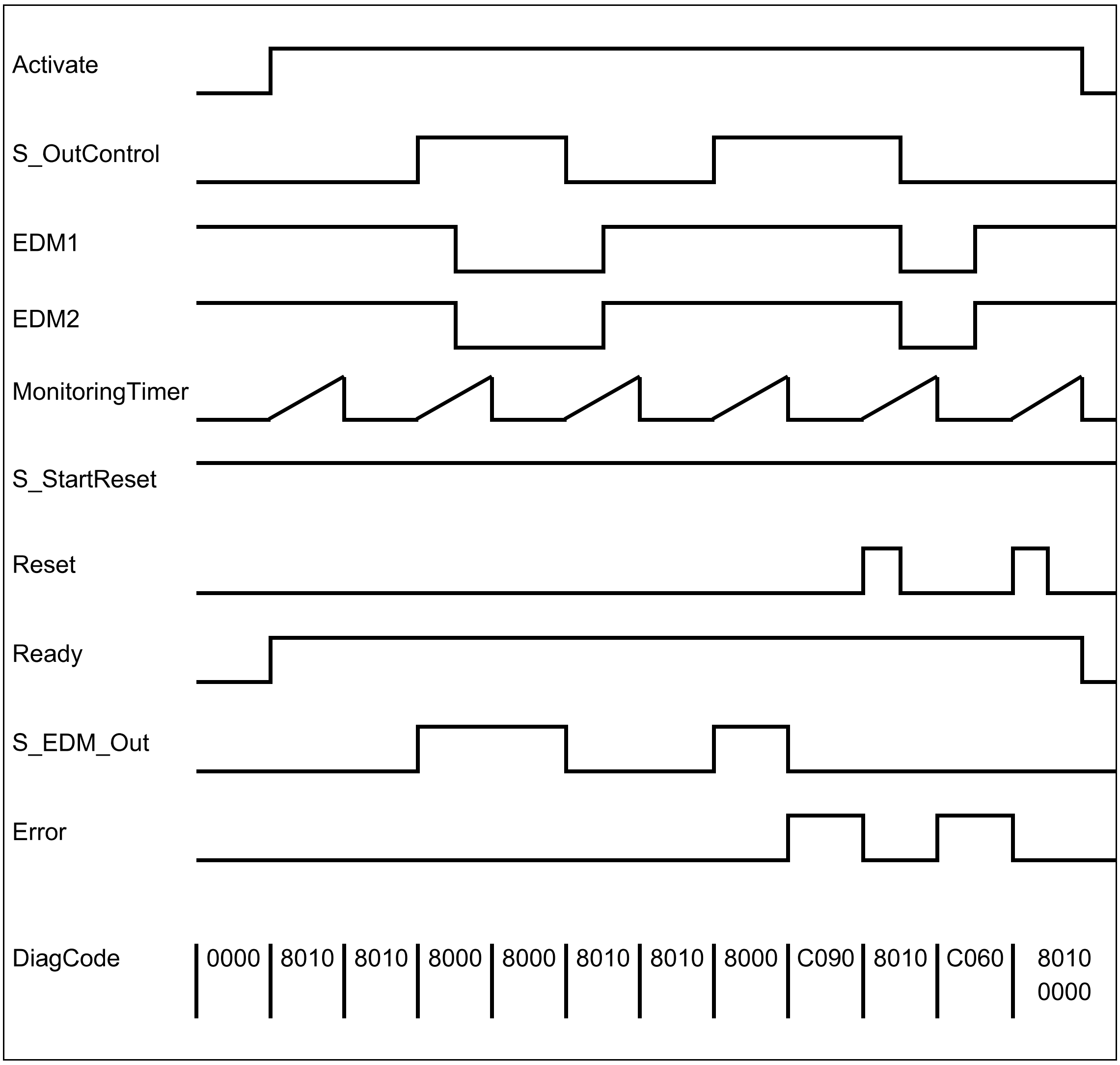


Fig. 120: Timing diagrams for SF_EDM: S_StartReset = TRUE

The following conditions force a transition to the error state:

- Invalid static Reset signal in the process.
- Invalid EDM signal in the process.
- S_OutControl and Reset are incorrectly interconnected due to programming error.

Error behavior

In error states, the outputs are as follows:

- In the event of an error, the S_EDM_Out is set to FALSE and remains in this safe state.
- An EDM error message must always be reset by a rising trigger at Reset.
- A Reset error message can be reset by setting Reset to FALSE.

After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

Function block-specific error and status codes

Table 70: FB-specific error codes

DiagCode	State name	State description and output setting
C001	Reset Error 1	Static Reset signal in state 8001. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C011	Reset Error 21	Static Reset signal or same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C010. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C021	Reset Error 22	Static Reset signal or same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C020. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C031	Reset Error 23	Static Reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C030. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C041	Reset Error 31	Static Reset signal or same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C040. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C051	Reset Error 32	Static Reset signal or same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C050. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C061	Reset Error 33	Static Reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C060. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C071	Reset Error 41	Static Reset signal in state C070. Ready = TRUE S_EDM_Out = FALSE Error = TRUE

DiagCode	State name	State description and output setting
C081	Reset Error 42	Static Reset signal in state C080. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C091	Reset Error 43	Static Reset signal in state C090. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C010	EDM Error 11	The signal at EDM1 is not valid in the initial actuator state. In state 8010 the EDM1 signal is FALSE when enabling S_OutControl. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C020	EDM Error 12	The signal at EDM2 is not valid in the initial actuator state. In state 8010 the EDM2 signal is FALSE when enabling S_OutControl. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C030	EDM Error 13	The signals at EDM1 and EDM2 are not valid in the initial actuator states. In state 8010, the EDM1 and EDM2 signals are FALSE when enabling S_OutControl. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C040	EDM Error 21	The signal at EDM1 is not valid in the initial actuator state. In state 8010, the EDM1 signal is FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C050	EDM Error 22	The signal at EDM2 is not valid in the initial actuator state. In state 8010, the EDM2 signal is FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C060	EDM Error 23	The signals at EDM1 and EDM2 are not valid in the initial actuator states. In state 8010, the EDM1 and EDM2 signals are FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE

DiagCode	State name	State description and output setting
C070	EDM Error 31	The signal at EDM1 is not valid in the actuator switching state. In state 8000, the EDM1 signal is TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C080	EDM Error 32	The signal at EDM2 is not valid in the actuator switching state. In state 8000, the EDM2 signal is TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C090	EDM Error 33	The signals at EDM1 and EDM2 are not valid in the actuator switching state. In state 8000, the EDM1 and EDM2 signals are TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C111	Init Error	Similar signals at S_OutControl and Reset (R_TRIG at same cycle) detected (may be a programming error). Ready = TRUE S_EDM_Out = FALSE Error = TRUE

Table 71: FB-specific status codes (no error):

DiagCode	State name	State description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EDM_Out = FALSE Error = FALSE
8001	Init	Block activation startup inhibit is active. Reset required. Ready = TRUE S_EDM_Out = FALSE Error = FALSE
8010	Output Dis- able	EDM control is not active. Timer starts when state is entered. Ready = TRUE S_EDM_Out = FALSE Error = FALSE
8000	Output Enable	EDM control is active. Timer starts when state is entered. Ready = TRUE S_EDM_Out = TRUE Error = FALSE

4.6.5 SafetyDeviceExt_LV100_PROFIsafe_AC500_V27.lib

This library includes a PROFIsafe F-Device stack implementation (PROFISAFEDEVICESTACK POU), which is a main F-Device component.

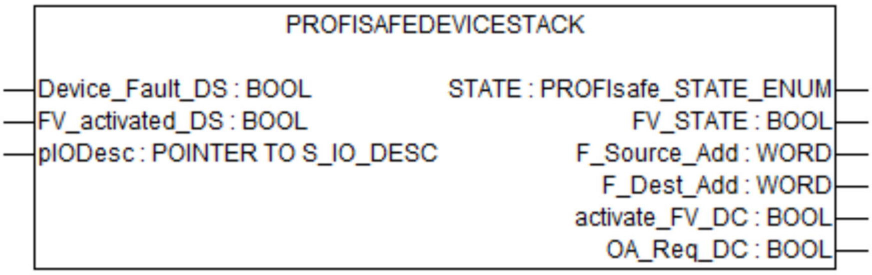


Table 72: FB name: PROFISAFEDEVICESTACK

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
Device_Fault_DS	BOOL	FALSE	Failure in device. This parameter allows the application to inform the F-Host about a malfunction. If Device_Fault_DS is set, the master stack sets FV_activated = 1 in the control byte.
FV_activated_DS	BOOL	FALSE	Fail-safe values activated. It allows the application to inform the F-Host that it uses fail-safe values. It is set internally by the PROFIsafe device stack when SM560-S-FD-1 / SM560-S-FD-4 is in DEBUG STOP state.
pIODesc	POINTER	NULL	Internal input parameter. **(Internal use only!)**
VAR_OUTPUT			
STATE	PROFIsafe_STATE_ENUM	PROFIsafe_STATE_ INIT	This parameter returns the current state of the PROFIsafe device stack. For example, the user can find out why the currently transmitted F-Parameter set was not accepted ➡ *Table 73 “PROFIsafe F-Device states” on page 303*.
FV_STATE	BOOL	TRUE	If TRUE, this parameter indicates that the device stack is delivering fail-safe value "0" to the F-Host program for every input value. Otherwise, process values are delivered.
F_Source_Add	WORD	0	This parameter represents the F-Source address that was transferred from the F-Host to this F-Device via the F-Parameters.
F_Dest_Add	WORD	0	This parameter specifies the F-Destination address, which shall match the switch address setting of SM560-S-FD-1 / SM560-S-FD-4 and the formula for the F-Destination addresses ➡ *Table 9 “F-Parameters of AC500-S safety modules” on page 142*.

Name	Data type	Initial value	Description, parameter values
activate_FV_DC	BOOL	FALSE	**This parameter is for debugging purposes only.** If TRUE, this parameter indicates to the F-Device that FV shall be used.
OA_Req_DC	BOOL	FALSE	**This parameter is for debugging purposes only.** If TRUE, the F-Host requests an operator acknowledgment for the F-Device from the F-Host safety application. In the event of an error (watchdog timeout or CRC, etc.) the fail-safe values are activated. If the error is no longer present (the communication with the module was re-established) and an operator acknowledgment is possible, the F-Host driver sets OA_Req_S = TRUE. If the F-Host application sets OA_C = TRUE, OA_Req_S is reset to FALSE and normal operation is resumed.

NOTICE!
Since the F-Device instances do not support iParameters, the function block has no possibility to set the bit iPar_OK_S in status byte or read the bit iPar_EN_C from the PROFIsafe control byte.

The PROFIsafe F-Device instances start asynchronously after power-up. F-Parameters are written to the PROFINET IO device (CM589-PNIO or CM589-PNIO-4) by the corresponding F-Host / PROFINET IO controller. F-Parameters are then transferred via the non-safety CPU to the SM560-S-FD-1 / SM560-S-FD-4, which can use them to parameterize F-Device instance.

If parameterization is repeated, F-Device instances are to be re-initialized at runtime. F-Parameters are only transferred by AC500 communication modules and non-safety CPU and are protected against transmission errors by the F_Par_CRC.

The F-Source address of an F-Device instance is set at runtime by the F-Host using the F_Source_Add parameter in F-Parameters. On SM560-S-FD-1 / SM560-S-FD-4, in addition to the normal tests of the F-Device stack, it is checked that the F-Source address of an F-Device instance does not overlap with the F-Source addresses of the own F-Host. If there is an overlap, the error is set for the newly parameterized F-Device instance.

As soon as the F-Device instance is configured, it continues to check that the F-Source addresses reported by the F-Host are valid. If not, the error is set and the boot project is not loaded.

The F-Device stack can report the following errors to the F-Host via the status byte:

- Device_Fault: malfunction in the device. This error can be triggered from the application using the Device_Fault_DS flag on the PROFISAFEDEVICESTACK FB.
- CE_CRC (communication error): CRC error or wrong consecutive number. This error is automatically triggered by the stack.
- WD_timeout (watchdog timeout): No valid PROFIsafe telegram received within the F_WD_Time. This error is automatically triggered by the stack.
- FV_activated_S (fail-safe values are activated): Indicates to the F-Host that FV are used. It can also be set by the FV_activated_DS flag from the F-Device application.

The F-Host can also detect communication errors (watchdog timeout, CRC error or incorrect consecutive number). The application behind the corresponding F-Device can be informed about these errors via the activate_FV_DC flag = TRUE of the PROFISAFEDEVICESTACK instance and can react accordingly.

The application can use the output variable "STATE" to obtain information about the current status of the F-Device instance.

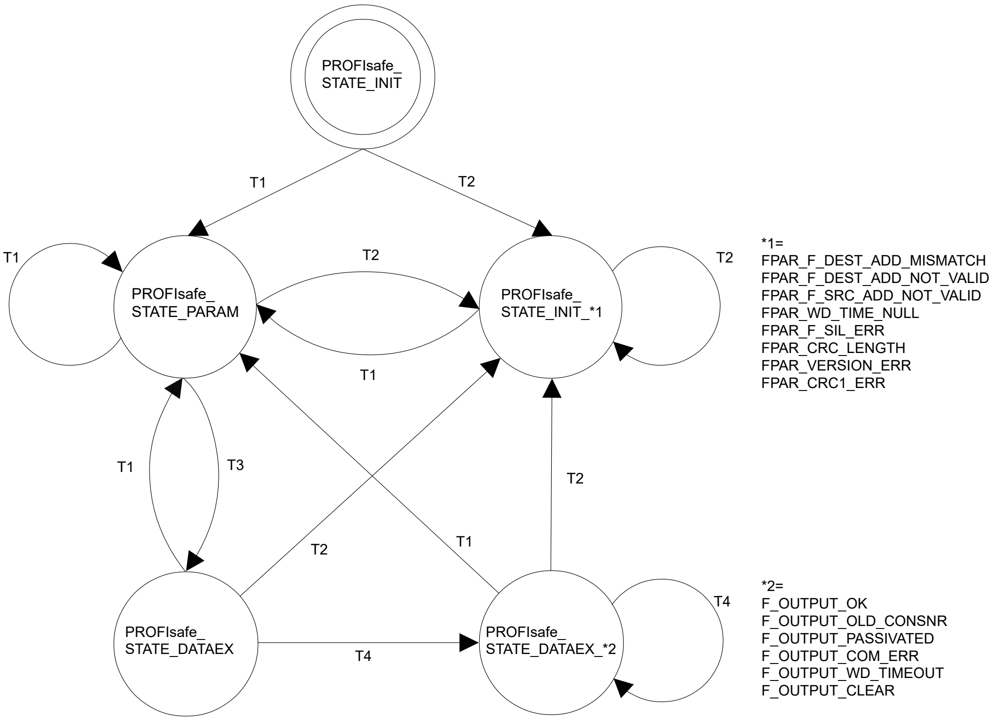


Fig. 121: PROFIsafe F-Device state diagram

T1 Good F-Parameters received
T2 Bad F-Parameters received
T3 F-Host limit not reached
T4 Message processed

The state transitions T1 and T2 are executed immediately when new F-Parameters have been transferred for the F-Device instance. If the F-Source address limit for the SM560-S-FD-1 (max. 1 F-Source address) / SM560-S-FD-4 (max. 4 different F-Source addresses) is not yet reached, transition T3 switches immediately. If the F-Source address limit has been reached, active F-Device instances (PROFIsafe_STATE_DATAEX states) of an F-Host must be stopped by T1 or T2 transition.

The following table describes the meaning of each state:

Table 73: PROFIsafe F-Device states

Value of STATE output on PROFIsafe F-Device stack instance	**Meaning**
PROFIsafe_STATE_INIT	Status after initialization of F-Device instances.
PROFIsafe_STATE_FPAR_F_DEST_ADD_MISMATCH	Parameterization error: F-Destination address does not correspond to the given value based on rotary address switch value on SM560-S-FD-1 / SM560-S-FD-4 safety CPU. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 28
PROFIsafe_STATE_FPAR_F_DEST_ADD_NOT_VALID	Parameterization error: F-Destination address invalid. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 1

Value of STATE output on PROFIsafe F-Device stack instance	Meaning
PROFIsafe_STATE_FPAR_F_SRC_ADD_NOT_VALID	Parameterization error: F-Source address is invalid or overlapping with F-Source addresses of F-Host instances. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 2
PROFIsafe_STATE_FPAR_WD_TIME_NULL	Parameterization error: Watchdog time set to zero. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 11
PROFIsafe_STATE_FPAR_F_SIL_ERR	Parameterization error: Requested SIL is too high. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 10
PROFIsafe_STATE_FPAR_CRC_LENGTH	Parameterization error: Required CRC length does not fit to the data length. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 42
PROFIsafe_STATE_FPAR_VERSION_ERR	Parameterization error: PROFIsafe version error. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 40
PROFIsafe_STATE_FPAR_CRC1_ERR	Parameterization error: CRC error in F-Parameters. Refer also to diagnosis *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* Module 28, Error 19
PROFIsafe_STATE_PARAM	F-Host limitation error: F-Parameters accepted, but the F-Device does not exchange data because of the F-Host limitation. No diagnosis message is available. If required, customized AC500 diagnosis message shall be generated.
PROFIsafe_STATE_DATAEX	F-Parameters are accepted, F-Device instance can exchange process data.
PROFIsafe_STATE_DATAEX_F_OUTPUT_OK	The PROFIsafe output telegram for F-Host is valid.
PROFIsafe_STATE_ DATAEX_F_OUTPUT_OLD_CONSNR	The PROFIsafe output telegram for F-Host is valid with an old consecutive number.
PROFIsafe_STATE_ DATAEX_F_OUTPUT_PASSIVATED	Communication error was detected or the F-Host sends "activate_FV" in PROFIsafe control byte. If required, customized AC500 diagnosis message shall be generated from the application (if PROFIsafe_STATE_ DATAEX_F_OUTPUT_PASSIVATED is detected on STATE output of F-Device stack instance).

Value of STATE output on PROFIsafe F-Device stack instance	Meaning
PROFIsafe_STATE_DATAEX_F_OUTPUT_COM_ERR	PROFIsafe error: CRC error in PROFIsafe output telegram is detected. If required, customized AC500 diagnosis message shall be generated from the application (if PROFIsafe_STATE_DATAEX_F_OUTPUT_COM_ERR is detected on STATE output of F-Device stack instance).
PROFIsafe_STATE_DATAEX_F_OUTPUT_WD_TIMEOUT	PROFIsafe error: Watchdog timeout detected. If required, customized AC500 diagnosis message shall be generated from the application (if PROFIsafe_STATE_DATAEX_F_OUTPUT_WD_TIMEOUT is detected on STATE output of F-Device stack instance).
PROFIsafe_STATE_DATAEX_F_OUTPUT_CLEAR	Zero telegram received, old PROFIsafe output data is still valid.

4.6.6 SafetyExt2_LV100_AC500_V27.lib

SafetyExt2_LV100_AC500_V27.lib library includes the following POUs:

System commands

- SF_SAFE_STOP (Triggering of the SAFE STOP on the safety CPU)

System information

- SF_MAX_POWER_DIP_GET_CFG (Getting the configured number of restarts after power dip in the safety CPU)
- SF_BOOTPROJECT_CRC (Getting boot project CRC)

4.6.6.1 SF_SAFE_STOP

The function SF_SAFE_STOP allows the user setting the safety CPU directly into the SAFE STOP state.

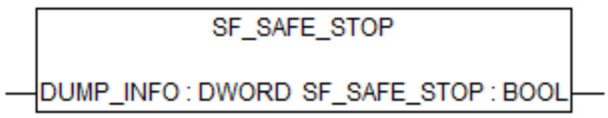


Table 74: FB name: SF_SAFE_STOP

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
DUMP_INFO	DWORD	16#00000000	The value DUMP_INFO is written to the core dump so that the user can find out together with the ABB support team at which point in his safety application the SAFE STOP state was triggered.
VAR_OUTPUT			
SF_SAFE_STOP	BOOL	FALSE	The output is not used and only available because functions must be defined with a return value. The application will not be able to evaluate the output as the safety CPU switches to the safe state.

Call in ST

```
SF_SAFE_STOP(DUMP_INFO:=16#B5006BB1);
```

4.6.6.2 SF_MAX_POWER_DIP_GET_CFG

The SF_MAX_POWER_DIP_GET_CFG function returns the configured maximum power dip value of the safety CPU ↬ *Chapter 4.6.7.2 “SF_MAX_POWER_DIP_SET” on page 308* ↬ *Chapter 4.6.7.6 “SF_MAX_POWER_DIP_GET” on page 312*.

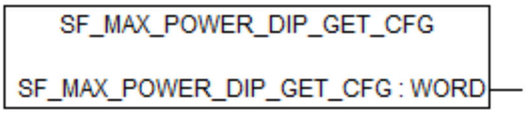


Table 75: FB name: SF_MAX_POWER_DIP_GET_CFG

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
SF_MAX_POWER_DIP_GET_CFG	WORD	16#0000	Configured maximum number of tolerated power dips (undervoltage/overvoltage faults).

Call in ST

```
MAX_POWER_DIPS_CFG := SF_MAX_POWER_DIP_GET_CFG();
```

4.6.6.3 SF_BOOTPROJECT_CRC

The SF_BOOTPROJECT_CRC function returns the CRC of the boot project which was in the flash memory when the safety CPU was started (it corresponds to the boot project CRC which is displayed in CODESYS Safety under the menu item *“Online ➔ Check bootproject in PLC”*).

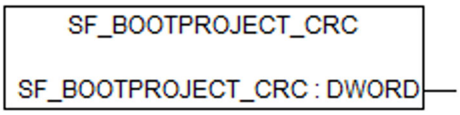


Table 76: FB name: SF_BOOTPROJECT_CRC

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
SF_BOOTPROJECT_CRC	DWORD	16#00000000	CRC of the boot project in flash memory when the Safety CPU was started.

Call in ST

```
BOOTPROJECT_CRC := SF_BOOTPROJECT_CRC();
```

4.6.7 SafetyExt_AC500_V22.lib

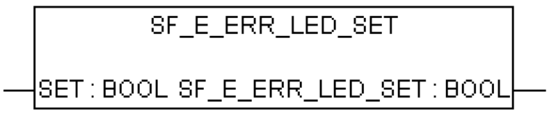
SafetyExt_AC500_V22.lib library includes the following POUs:

- System commands
  – SF_E_ERR_LED_SET (Setting E-ERR LED state (ON or OFF))
  – SF_MAX_POWER_DIP_SET (Setting the maximum number of restarts after power dip in the safety CPU)
  – SF_WDOG_TIME_SET (Setting the maximum allowed cycle time of the safety CPU)
  – SF_APPL_MEASURE_BEGIN (This function defines the start point of time profiling)
  – SF_APPL_MEASURE_END (This function defines the end point of time profiling)
- System information
  – SF_MAX_POWER_DIP_GET (Getting the current number of restarts after power dip in the safety CPU)
  – SF_SAFETY_MODE (Reading out if the safety CPU is in DEBUG or SAFETY mode)
  – SF_SM5XX_OWN_ADR (Getting the value of the hardware switch address on the safety CPU)
  – SF_RTS_INFO (It provides the firmware version of the safety CPU. The version is a binary coded decimal, e.g., 16#10 means version 1.0)
- Data storage
  – SF_FLASH_DEL (This function block deletes a data segment in the flash memory. All data in this data segment will be deleted.)
  – SF_FLASH_READ (The function block reads a data set from a data segment of the flash memory and stores the read data set beginning at the start flag defined by the safety CPU.)
  – SF_FLASH_WRITE (The function block writes data to a data segment in the flash memory.)
- Acyclic non-safe data exchange
  – SF_DPRAM_PM5XX_S_REC (Receiving data from non-safety CPU)
  – SF_DPRAM_PM5XX_S_SEND (Sending data to non-safety CPU)

NOTICE!

For establishing an acyclic non-safe data exchange between safety and non-safety CPU, you have to use dedicated function blocks for the non-safety CPU ➪ *Appendix B.5.1 “Acyclic non-safe data exchange” on page 385* ➪ *Appendix C.5.1 “Acyclic non-safe data exchange” on page 401*.

4.6.7.1 SF_E_ERR_LED_SET



Setting E-ERR LED state (ON = TRUE or OFF = FALSE)

E-ERR LED is set directly in the same safety CPU cycle. The state remains unchanged until it is not explicitly changed using SF_E_ERR_LED_SET call.

Table 77: FUN name: SF_E_ERR_LED_SET

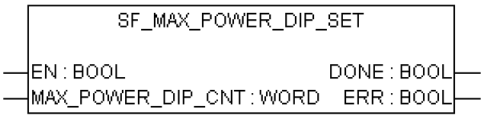
Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
SET	BOOL	FALSE	FALSE = E-ERR LED is OFF, TRUE = E-ERR LED is ON

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
SF_E_ERR_LED_SET	BOOL	FALSE	FALSE = E-ERR LED is OFF, TRUE = E-ERR LED is ON

Call in ST

```
SF_E_ERR_LED_SET_Value := SF_E_ERR_LED_SET(SF_E_ERR_LED_SET_Set);
```

4.6.7.2 SF_MAX_POWER_DIP_SET



Setting the maximum number of power dips in SM560-S safety CPU

The SF_MAX_POWER_DIP_SET function block allows users to control the number of power cycles to successfully restart the safety CPU in RUN mode after the power dip was detected. One or two power cycles may be required, depending on the power dip detection state, as it is further explained in the text. The function block provides an input MAX_POWER_DIP_CNT for the maximum number of warm starts after power dips. The warm start of the safety CPU after power dip means that only one power cycle or *"reboot"* after power dip detection is needed to restart the safety CPU. The cold start of the safety CPU after power dip requires two power cycles or *"reboot"* twice after power dip detection. Normally, a cold start will be needed, if SF_MAX_POWER_DIP_SET is not used.

To avoid repeated power dip detection on the safety CPU, make sure that the power-off phase of the power cycle lasts for at least 1.5 s before the power-on is performed.

If this function block is not available in the safety application program or not activated, then the initial value (= "0") is stored in the flash memory instead. The number of warm starts that occurred after power dips is counted and compared to the number available prior to the start of the safety application program. If the number is higher, then the application on the safety CPU is not executed after 1 power cycle. This condition can only be terminated by switching the voltage off and then powering the safety CPU on again after a pause of at least 1.5 s.

Only one function block instance must be used in the safety program, otherwise a warning is issued.

> **!**
>
> **NOTICE!**
>
> Each time SF_MAX_POWER_DIP_SET FB is called with EN transition from FALSE to TRUE, the internal power dip counter value is reset, which means that power dip counter will be started from 0 now. Thus, it makes sense to call SF_MAX_POWER_DIP_SET FB in safety program only once with EN transition from FALSE to TRUE as a one-time parameterization of power dip functionality.
>
> If you do not follow the recommendation above, each time SF_MAX_POWER_DIP_SET FB is called with EN transition from FALSE to TRUE in the safety application program, the counter value for restarts after power dip in the safety CPU, which can be read from SF_MAX_POWER_DIP_GET FB, will be reset to '0'.

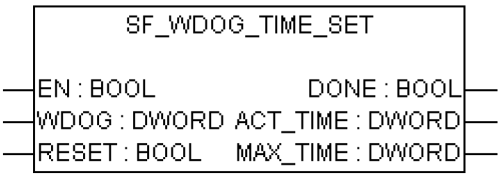
Table 78: FB name: SF_MAX_POWER_DIP_SET

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	The block is activated to store MAX_POWER_DIP_CNT value in the flash memory using a transition of EN input from FALSE to TRUE. The block remains active and ignores any changes on EN input until DONE output is equal to TRUE. The MAX_POWER_DIP_CNT value can be stored in the flash memory only if the transition on EN input from FALSE to TRUE is triggered.
MAX_POWER_DIP_CNT	WORD	16#0000	Maximum number of the safety CPU warm starts after power dip errors from power supply of non-safety CPU. Warm start means that only one power cycle after power dip detection is needed to restart the safety CPU.
VAR_OUTPUT			
DONE	BOOL	FALSE	Output DONE indicates that the set process is finished (see also ERR output).
ERR	BOOL	FALSE	If TRUE, then error occurred during the set process (saving of MAX_POWER_DIP_CNT value to the flash memory).

Call in ST

```
SF_MAX_POWER_DIP_SET (EN := SF_MAX_POWER_DIP_SET_EN,
MAX_POWER_DIP_CNT := SF_MAX_POWER_DIP_SET_MAX_POWER_DIP_CNT,
DONE => SF_MAX_POWER_DIP_SET_DONE, ERR => SF_MAX_POWER_DIP_SET_ERR);
```

4.6.7.3 SF_WDOG_TIME_SET



Setting the maximum allowed cycle time of the safety CPU

The SF_WDOG_TIME_SET function block allows the user to monitor the cycle time. The function block must be called by the user during the first cycle. In order to update the outputs ACT_TIME and MAX_TIME, it is necessary to call the function block in each cycle. If the function block is not available in the application, the safety CPU and the application program will enter the SAFE STOP state after the first cycle. The watchdog time is monitored prior to the output of the PROFIsafe telegrams.

If the cycle time is exceeded, an error message is an output and the safety CPU enters the SAFE STOP state. Reasonable values are longer than the typical safety CPU runtime and at least two times shorter than the F_WD_Time of the safety I/O module.

Only one function block instance must be used in the safety program, otherwise a warning is issued.

> **NOTICE!**
> The cycle time supervision takes place only in RUN (safety) mode.

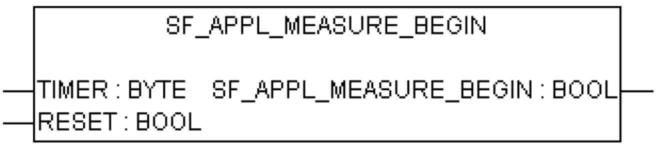
Table 79: FB name: SF_WDOG_TIME_SET

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	The function block is activated (EN = TRUE) or deactivated (EN = FALSE) via input EN. If the block is active, the current values are available at the outputs.
WDOG	DWORD	16#00000000	Watchdog time in ms. The maximum allowed value is 1000. If WDOG is > 1000, then SAFE STOP state will be entered by the safety CPU.
RESET	BOOL	FALSE	TRUE sets MAX_TIME to 0.
VAR_OUTPUT			
DONE	BOOL	FALSE	Output DONE indicates that the set process is finished.
ACT_TIME	DWORD	16#00000000	Actual safety CPU cycle time in ms
MAX_TIME	DWORD	16#00000000	Maximal monitored safety CPU cycle time in ms

Call in ST

```
SF_WDOG_TIME_SET (EN := SF_WDOG_TIME_SET_EN,
WDOG := SF_WDOG_TIME_SET_WDOG,
RESET := SF_WDOG_TIME_SET_RESET,
DONE => SF_WDOG_TIME_SET_DONE,
ACT_TIME => SF_WDOG_TIME_SET,
MAX_TIME => SF_WDOG_TIME_SET_MAX_TIME);
```

4.6.7.4 SF_APPL_MEASURE_BEGIN



Defining the start point of time profiling

This function defines the start point of time profiling within safety application program and shall be used together with SF_APPL_MEASURE_END function. The time profiling results can be seen only using "applinfo" PLC browser command and cannot be used within safety application program.

The time between the calls of SF_APPL_MEASURE_BEGIN and SF_APPL_MEASURE_END functions in the safety application program is measured (including within one safety CPU cycle) and saved in the timer identified with the value set for input parameter TIMER.

> **NOTICE!**
> SF_APPL_MEASURE_BEGIN function was developed for measuring short time intervals only, which means that for time intervals of ~ 10 minutes and longer, it produces invalid results.

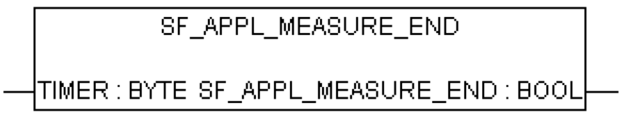
Table 80: FUN name: SF_APPL_MEASURE_BEGIN

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
TIMER	BYTE	16#00	Timer identification. The allowed range is from 0 to 31.
RESET	BOOL	FALSE	If TRUE, then MAX and MIN results of time profiling will be deleted. Otherwise, the observed values are kept.
VAR_OUTPUT			
SF_APPL_MEASURE_BEGIN	BOOL	FALSE	Return value is TRUE if the TIMER value is within the allowed range (0 ... 31), otherwise the return value is FALSE.

Call in ST

```
SF_APPL_MEASURE_BEGIN_VALUE :=
SF_APPL_MEASURE_BEGIN(SF_APPL_MEASURE_BEGIN_TIMER,
SF_APPL_MEASURE_BEGIN_RESET);
...
...
SF_APPL_MEASURE_END_VALUE :=
SF_APPL_MEASURE_END(SF_APPL_MEASURE_END_TIMER);
```

4.6.7.5 SF_APPL_MEASURE_END



Defining the end point of time profiling

This function defines the end point of time profiling within safety application program and shall be used together with SF_APPL_MEASURE_BEGIN function. The time profiling results can be seen only using "applinfo" PLC browser command and cannot be used within safety application program.

The time between the calls of SF_APPL_MEASURE_BEGIN and SF_APPL_MEASURE_END functions in the safety application program is measured and saved in the timer identified with the value set for input parameter TIMER.

> **NOTICE!**
> SF_APPL_MEASURE_END function was developed for measuring short time intervals only, which means that for time intervals of ~ 10 minutes and longer, it produces invalid results.

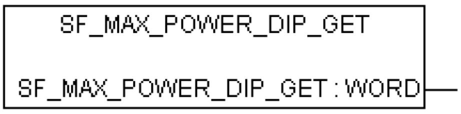
Table 81: FUN name: SF_APPL_MEASURE_END

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
TIMER	BYTE	16#00	Timer identification. The allowed range is from 0 to 31.
VAR_OUTPUT			
SF_APPL_MEASURE_END	BOOL	FALSE	Return value is TRUE if the TIMER value is within the allowed range (0 .. 31), otherwise the return value is FALSE.

Call in ST

```
SF_APPL_MEASURE_BEGIN_VALUE :=
SF_APPL_MEASURE_BEGIN(SF_APPL_MEASURE_BEGIN_TIMER,
SF_APPL_MEASURE_BEGIN_RESET);
...
...
SF_APPL_MEASURE_END_VALUE :=
SF_APPL_MEASURE_END(SF_APPL_MEASURE_END_TIMER);
```

4.6.7.6 SF_MAX_POWER_DIP_GET



Getting the current number of restarts after power dip in the safety CPU

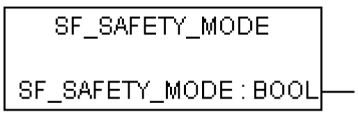
Table 82: FUN name: SF_MAX_POWER_DIP_GET

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
SF_MAX_POWER_DIP_GET	WORD	16#0000	Actual value of power dip error counter.

Call in ST

```
SF_MAX_POWER_DIP_GET_Value := SF_MAX_POWER_DIP_GET();
```

4.6.7.7 SF_SAFETY_MODE



Reading out if the safety CPU is in DEBUG RUN (non-safety), DEBUG STOP (non-safety) or in RUN (safety) mode

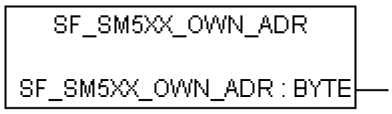
Table 83: FUN name: SF_SAFETY_MODE

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
SF_SAFETY_MOD E	BOOL	FALSE	Safety CPU mode: ● FALSE: DEBUG RUN (non-safety) or DEBUG STOP (non-safety) mode is active. ● TRUE: RUN (safety) mode is active.

Call in ST

```
SF_SAFETY_MODE_Value := SF_SAFETY_MODE();
```

4.6.7.8 SF_SM5XX_OWN_ADR



Getting the value of the hardware switch address on the safety CPU

Only the value set during SM560-S safety CPU start-up is read. Further changes of the hardware switch address are ignored.

> **NOTICE!**
> Despite the fact that SF_SM5XX_OWN_ADR function is a safety POU, the hardware switch address value is a non-safety value and needs additional measures to satisfy functional safety requirements.

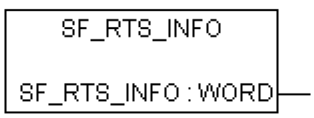
Table 84: FUN name: SF_SM5XX_OWN_ADR

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
SF_SM5XX_OWN_ ADR	BYTE	16#00	Value of the hardware switch address on the safety CPU set during its start-up.

Call in ST

```
SF_SM5XX_OWN_ADR_Value := SF_SM5XX_OWN_ADR();
```

4.6.7.9 SF_RTS_INFO



Display of the firmware version of the safety CPU

This function provides the firmware version of the safety CPU. The version is a binary coded decimal, e.g., 16#10 means version 1.0.

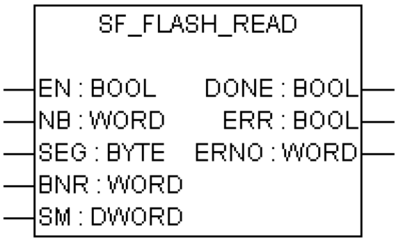
Table 85: FUN name: SF_RTS_INFO

Name	Data type	Initial value	Description, parameter values
VAR_OUTPUT			
SF_RTS_INFO	WORD	16#0000	Firmware version of the safety CPU. The upper BYTE of the entry represents the main version; the lower BYTE represents the subversion of the runtime system. Example: RTS_VERSION = 16#0110 ➔ V01.1.0

Call in ST

```
SF_RTS_INFO_Value := SF_RTS_INFO();
```

4.6.7.10 SF_FLASH_READ



Reading of user data from the flash memory

The function block reads a data set from a data segment in the flash memory and stores this data set beginning at the starting flag defined at input SM. The data contained in the data set were previously stored to the flash memory using the SF_FLASH_WRITE function block.

> **NOTICE!**
> Access to the flash memory is only possible using the function blocks SF_FLASH_WRITE, SF_FLASH_DEL and SF_FLASH_READ.
>
> NB blocks are read starting at block BNR within segment SEG and stored starting at address SM.
>
> 32 binary data or 16 word data or 8 double word data are read per block.
>
> One block contains 38 bytes:
>
> – 32 bytes of data
> – 4 bytes for CRC checksum
> – 1 byte as "written" identifier
> – 1 byte for alignment
>
> *➪ Table 87 “Structure of one of the flash memory segments with user data” on page 316*
>
> Reading a data set is triggered once by a FALSE/TRUE edge at input EN. If no error occurred while reading the data, output DONE is set to TRUE and the outputs ERR and ERNO are set to FALSE. The data set is stored beginning at the defined start flag SM.
>
> Storing the data set can take several CPU cycles.
>
> If an error occurs during reading, DONE and ERR are set to TRUE and data from SM are equal to 0. The error type is indicated at output ERNO.

> **NOTICE!**
> This function block is activated by a positive edge of the input variable EN. During the cycle where the function block notices that the operation is finished (output DONE = TRUE) it will set the output variables only for one cycle. When the function block is called again it will reset the output variables immediately.

Table 86: FB name: SF_FLASH_READ

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	Activation of the FB using a positive edge The following applies: ● EN = FALSE/TRUE edge: Reading the data set is carried out once. ● EN = TRUE: The function block is not processed, i.e. it does not change its outputs anymore.
NB	WORD	16#0000	Number of data set blocks (decimal 1 ... 1724) Input NB is used to specify the number of blocks contained in the data set. 32 byte data or 16 word data or 8 double word data are read per block. Valid values: 1 ... 1724 Example: ● SM = ADR(%MW0.0) and NB = 1: Data are stored at %MW0.0 to %MW0.15 (1 block = 16 word data) ● SM = ADR(%MW0.0) and NB = 2: Data are stored at %MW0.0 to %MW0.31 (2 blocks = 32 word data)
SEG	BYTE	16#00	ID number of the data segment (16#01 or 16#02)
BNR	WORD	16#0000	Starting block number in the flash memory data segment (decimal 0 ... 1723)
SM	DWORD	16#00000000	Destination address for the read data set (address of the first variable where the data are placed)
VAR_OUTPUT			
DONE	BOOL	FALSE	Reading procedure is completed (DONE = TRUE) This output always has to be considered together with output ERR. The following applies: ● DONE = TRUE and ERR = FALSE: Reading completed. The data set has been stored beginning at the defined input SM. ● DONE = TRUE and ERR = TRUE: An error occurred while reading the data set. Output ERNO indicates the error number.

Name	Data type	Initial value	Description, parameter values
ERR	BOOL	FALSE	Error occurred (data segment could not be read) This output always has to be considered together with output DONE. The following applies if an error occurred: DONE = TRUE and ERR = TRUE. Output ERNO indicates the error number.
ERNO	WORD	16#0000	Error number ↬ *[4]*. Output ERNO indicates an error number. This output always has to be considered together with the outputs DONE and ERR. The SF_FLASH_READ operation may take quite a long time since the safety CPU user program is processed with higher priority. Output ERNO indicates that the function block has started the execution (0x0FFF = BUSY). During this phase, the outputs ERR and DONE are set to FALSE.

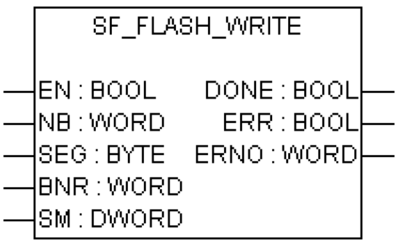
Table 87: Structure of one of the flash memory segments with user data

Byte:		**1 | 2**	**3 | 4**	**5 | 6**	**...**	**29 | 30**	**31 | 32**	**33 ... 36**	**37**	**38**
Byte offset	**Block no.**	**Word 1**	**Word 2**	**Word 3**	**...**	**Word 15**	**Word 16**	**CRC**	**Written identifier**	**Alignment**
0	0									
38	1									
76	2									
...	...									
65436	1722									
65474	1723									

Call in ST

```
READ_FLASH(EN := EN_FLASH_READ,
NB := NB_FLASH_READ,
SEG := SEG_FLASH_READ,
BNR := BNR_FLASH_READ,
SM := SM_FLASH_READ,
DONE => DONE_FLASH_READ,
ERR => ERR_FLASH_READ,
ERNO => ERNO_FLASH_READ);
```

4.6.7.11 SF_FLASH_WRITE



Writing of user data to the flash memory

The function block writes a data set to a data segment in the flash memory. For that purpose, two data segments are available in the safety CPU. The delete operation (function block SF_FLASH_DEL) always deletes a data segment as a whole. One data segment consists of 1724 blocks (0 ... 1723). Each block comprises 38 bytes. The maximum number of writing cycles to the flash memory is limited. Deleting data in the flash memory is also considered to be a "writing" cycle.

After a delete operation, data can be written only once to each of these 1724 data segment blocks. If a block containing data is to be overwritten with new data, the entire data segment has to be deleted first. In doing so, all data in this segment will be lost.

NB blocks are read starting at address SM and stored in segment SEG starting at block BNR.

32 binary data or 16 word data or 8 double word data are read per block.

One block contains 38 bytes:

- 32 bytes of data
- 4 bytes for CRC checksum
- 1 byte as "written" identifier
- 1 byte for alignment

Table 87 "Structure of one of the flash memory segments with user data" on page 316

Once the write operation for a data set has been started (by a FALSE/TRUE edge at input EN), the data contained in the data set must not be changed anymore until the write operation completes (DONE = TRUE). Storing the data set in the flash memory can take several safety CPU cycles.

With a FALSE/TRUE edge at input EN, the data set is written once. Input EN is not evaluated again until the storage procedure is finished (DONE = TRUE).

After the write operation is finished, the function block outputs DONE, ERR and ERNO are updated. Data storage was successful if DONE = TRUE and ERR = FALSE. If DONE = TRUE and ERR = TRUE, an error occurred. The error type is indicated at output ERNO.

A new FALSE/TRUE edge at input EN starts a new write operation. Input BNR must point to the next free block for the next write operation since no new data can be written to blocks that already contain data without a preceding deletion of the data segment.

> **NOTICE!**
> This function block is activated by a positive edge of the input variable EN. During the cycle where the function block notices that the operation is finished (output DONE = TRUE) it will set the output variables only for one cycle. When the function block is called again it will reset the output variables immediately.

Table 88: FB name: SF_FLASH_WRITE

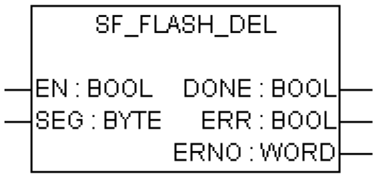
Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	Activation of the FB using a positive edge The following applies: ● EN = FALSE/TRUE edge: Reading the data set is carried out once. ● EN = TRUE: The function block is not processed, i.e. it does not change its outputs anymore.

Name	Data type	Initial value	Description, parameter values
NB	WORD	16#0000	Number of data set blocks (decimal 1 .. 1724) Input NB is used to specify the number of blocks contained in the data set. 32 byte data or 16 word data or 8 double word data are read per block. Valid values: 1 ... 1724 Example: - SM = ADR(%MW0.0) and NB = 1: Data are stored at %MW0.0 to %MW0.15 (1 block = 16 word data) - SM = ADR(%MW0.0) and NB = 2: Data are stored at %MW0.0 to %MW0.31 (2 blocks = 32 word data)
SEG	BYTE	16#00	ID number of the data segment (16#01 or 16#02)
BNR	WORD	16#0000	Starting block number in the flash memory data segment (decimal 0 ... 1723)
SM	DWORD	16#00000000	Source start address (address of the first variable from where the data will be written to the flash memory) At input SM, the address of the first variable of the data set is specified using an ADR operator. Once the write operation for a data set has been started (by a FALSE/TRUE edge at input EN), the data contained in the data set must not be changed anymore until the write operation is finished (DONE = TRUE).
VAR_OUTPUT			
DONE	BOOL	FALSE	Writing procedure is completed (DONE = TRUE) This output always has to be considered together with output ERR. The following applies: ● DONE = TRUE and ERR = FALSE: Write operation completed. The data set has been stored in the flash. ● DONE = TRUE and ERR = TRUE: An error occurred during the write operation. Output ERNO indicates the error number.
ERR	BOOL	FALSE	Error occurred (data segment could not be written) Output ERR indicates whether an error occurred during the write operation. This output always has to be considered together with output DONE. The following applies if an error occurred: DONE = TRUE and ERR = TRUE. Output ERNO indicates the error number.
ERNO	WORD	16#0000	Error number ➪ *[4]* Output ERNO indicates an error number. This output always has to be considered together with the outputs DONE and ERR. The SF_FLASH_WRITE operation may take quite a long time since the safety PLC user program is processed with higher priority. Output ERNO then indicates that the function block has started the execution (0x0FFF = BUSY). During this phase, the outputs ERR and DONE are set to FALSE.

Call in ST

```
WRITE_FLASH(EN := EN_FLASH_WRITE,
NB := NB_FLASH_WRITE,
SEG := SEG_FLASH_WRITE,
BNR := BNR_FLASH_WRITE,
SM := SM_FLASH_WRITE,
DONE => DONE_FLASH_WRITE,
ERR => ERR_FLASH_WRITE,
ERNO => ERNO_FLASH_WRITE);
```

4.6.7.12 SF_FLASH_DEL



Delete a selected segment from the flash memory

This function block deletes a selected segment with user data from the flash memory.

Input SEG defines the data segment within the flash memory. In the safety CPU, two segments numbered 1 and 2 (each providing 64 kB incl. CRC, flag and alignment) are reserved for the user. Deleting a data segment within the flash memory may take several PLC cycles.

Deletion of the data segment is triggered once by a FALSE/TRUE edge at input EN. Input EN will not be evaluated again until the delete operation is completed (DONE = TRUE).

After the deletion procedure is finished, all function block outputs are updated. The deletion was successful if DONE = TRUE and ERR = FALSE. If the outputs show DONE = TRUE and ERR = TRUE, the data segment could not be deleted.

> **NOTICE!**
> This function block is activated by a positive edge of the input variable EN. During the cycle where the function block notices that the operation is finished (output DONE = TRUE) it will set the output variables only for one cycle. When the function block is called again it will reset the output variables immediately.

Table 89: FB name: SF_FLASH_DEL

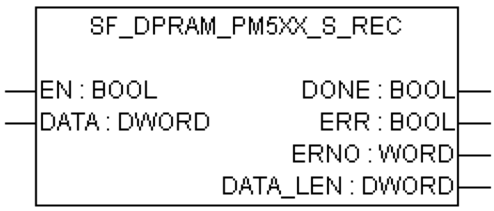
Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	Activation of the FB using a positive edge Deletion of the data segment is started once. Input EN will not be evaluated again until the delete operation is finished (DONE = TRUE). EN = TRUE: The function block is not processed, i.e. it does not change its outputs anymore. This is not valid during a delete operation.
SEG	BYTE	16#00	ID number of the data segment (16#01 or 16#02)
VAR_OUTPUT			

Name	Data type	Initial value	Description, parameter values
DONE	BOOL	FALSE	Delete procedure is completed (DONE = TRUE) Output DONE indicates that deletion of the data segment is completed. This output always has to be considered together with output ERR. The following applies: ● DONE = TRUE and ERR = FALSE: Deletion completed. The data segment has been deleted successfully. ● DONE = TRUE and ERR = TRUE: An error occurred while deleting the data segment. The data segment could not be deleted successfully.
ERR	BOOL	FALSE	Error occurred (data segment could not be deleted) Output ERR indicates whether an error occurred during deletion. This output always has to be considered together with output DONE. The following applies if the data segment could not be deleted: DONE = TRUE and ERR = TRUE. Output ERNO indicates the error number.
ERNO	WORD	16#0000	Error number *[4]*. Output ERNO indicates an error number. This output always has to be considered together with the outputs DONE and ERR. The SF_FLASH_DEL operation may take quite a long time since the safety CPU user program is processed with higher priority. Output ERNO indicates that the function block has started the execution (0x0FFF = BUSY). During this phase, the outputs ERR and DONE are set to FALSE.

Call in ST

```
DEL_FLASH(EN := EN_FLASH_DEL,
SEG := SEG_FLASH_DEL,
DONE => DONE_FLASH_DEL,
ERR => ERR_FLASH_DEL,
ERNO => ERNO_FLASH_DEL);
```

4.6.7.13 SF_DPRAM_PM5XX_S_REC



Reading the data from non-safety CPU to safety application on safety CPU

DANGER!
It is not recommended to transfer data values from non-safety CPU to safety CPU. If doing so, end-users have to define additional process-specific validation procedures in the safety program to check the correctness of the transferred non-safety data, if they would like to use those non-safety values for safety functions.

It is of no concern to transfer data values from safety CPU to non-safety CPU, e.g., for diagnosis and later visualization on operator panels.

DANGER!
If SF_DPRAM_PM5XX_S_REC function block is used to receive data from the non-safety CPU, then SIL 3 (IEC 61508 and IEC 62061) and PL e (ISO 13849-1) functional safety requirements will not be fulfilled for received data (independently on application safety communication profile used), because only one microprocessor (no 1oo2 safety architecture in the background) on safety CPU handles the receiving direction.

Contact ABB technical support on how to reach SIL 3 and PL e.

The SF_DPRAM_PM5XX_S_REC function block is used to receive data from the non-safety CPU. This data is stored in the memory area (DATA, memory address for received data, provided via ADR operator). The function block is enabled by a TRUE signal at input EN. It remains active until input EN is set to FALSE. Output DATA_LEN displays the length of the received data in bytes. DONE = TRUE and ERR = FALSE indicate successful data reception. If an error was detected during function block processing, the error is indicated at the outputs ERR and ERNO.

NOTICE!
Reception using the SF_DPRAM_SM5XX_S_REC function block is not edge-triggered. Therefore, input EN has to be continuously set to TRUE during data reception.

Table 90: FB name: SF_DPRAM_PM5XX_S_REC

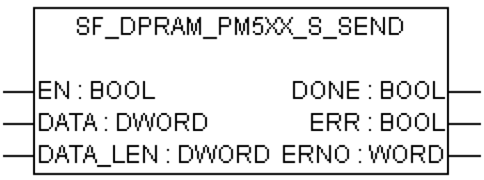
Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	Processing of this function block is controlled by input EN. The function block is active if EN = TRUE. The reception of data is indicated by output DONE.
DATA	DWORD	16#00000000	Input DATA is used to specify the address of the variable to which the user data is to be copied to. The address specified at DATA has to belong to a variable of the type ARRAY or STRUCT. Set the variable size to the maximum expected amount of data in order to avoid overlapping of memory areas.
VAR_OUTPUT			

Name	Data type	Initial value	Description, parameter values
DONE	BOOL	FALSE	Output DONE indicates the reception of data. This output always has to be considered together with output ERR. The following applies: ● DONE = TRUE and ERR = FALSE: Reception completed. A data set was received correctly. ● DONE = TRUE and ERR = TRUE: An error occurred during reception. The error number is indicated at output ERNO.
ERR	BOOL	FALSE	Output ERR indicates whether an error occurred during reception. This output always has to be considered together with output DONE. The following applies if an error occurred during reception: DONE = TRUE and ERR = TRUE. Output ERNO indicates the error number.
ERNO	WORD	16#0000	Error number ↳ *[4]*. Output ERNO provides an error identifier if an invalid value has been applied to an input or if an error occurred during job processing. ERNO always has to be considered together with the outputs DONE and ERR. The output value at ERNO is only valid if DONE = TRUE and ERR = TRUE.
DATA_LEN	DWORD	16#00000000	Output DATA_LEN displays the length of the received data in bytes (the maximum number is 84). The output value at DATA_LEN is only valid if DONE = TRUE.

Call in ST

```
PM5xxRec (EN := PM5xxRec_EN,
DATA := ADR(PM5xxRec_DATA),
DONE => PM5xxRec_DONE,
ERR => PM5xxRec_ERR,
ERNO => PM5xxRec_ERNO,
DATA_LEN => PM5xxRec_DATA_LEN);
```

4.6.7.14 SF_DPRAM_PM5XX_S_SEND



Sending data from the safety CPU to non-safety CPU

The SF_DPRAM_PM5XX_S_SEND function block is used to send data to the non-safety CPU. The data to be sent is available in the memory area (DATA, memory address for data to be transmitted, provided via ADR operator). The function block is activated with a TRUE signal (FALSE/TRUE edge) at input EN. The length of the data to be transmitted is specified in bytes at input DATA_LEN. DONE = TRUE and ERR = FALSE indicate that the sending process was successful. If an error was detected during function block processing, the error is indicated at the outputs ERR and ERNO.

DANGER!
If FB SF_DPRAM_PM5XX_S_SEND is used to send safety data from safety CPU to non-safety CPU, then SIL 3 (IEC 61508 and IEC 62061) and PL e (ISO 13849-1) functional safety requirements will not be fulfilled for sent data (independently on application safety communication profile used), because only one microprocessor (no 1oo2 safety architecture in the background) on safety CPU handles the sending direction.

Contact ABB technical support on how to reach SIL 3 and PL e.

NOTICE!
Sending data using the SF_DPRAM_PM5XX_S_SEND function block is edge-triggered, i.e. each sending process is initiated by a FALSE/TRUE edge at input EN.

NOTICE!
This function block is activated by a positive edge of the input variable EN. During the cycle where the function block notices that the operation is finished (output DONE = TRUE) it will set the output variables only for one cycle. When the function block is called again it will reset the output variables immediately.

Table 91: FB name: SF_DPRAM_PM5XX_S_SEND

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	Enabling of function block processing. Processing of this function block is controlled by input EN. The data transfer is initiated by a FALSE/TRUE edge. The sending of data is indicated by output DONE.
DATA	DWORD	16#00000000	Input DATA is used to specify the address of the variable the user data are to be copied to. The address specified at DATA has to belong to a variable of the type ARRAY or STRUCT. Set the variable size to the maximum expected amount of data in order to avoid overlapping of memory areas.
DATA_LEN	DWORD	16#00000000	The length of the data to be transmitted is specified in bytes at input DATA_LEN. The maximum number is 84.
VAR_OUTPUT			
DONE	BOOL	FALSE	Output DONE indicates the sending of data. This output always has to be considered together with output ERR. The following applies: ● DONE = TRUE and ERR = FALSE: Sending completed. A data set was sent correctly. ● DONE = TRUE and ERR = TRUE: An error occurred during sending. The error number is indicated at output ERNO.

Name	Data type	Initial value	Description, parameter values
ERR	BOOL	FALSE	Output ERR indicates whether an error occurred during sending. This output always has to be considered together with output DONE. The following applies if an error occurred during sending: DONE = TRUE and ERR = TRUE. Output ERNO indicates the error number.
ERNO	WORD	16#0000	Error number ↬ *[4]*. Output ERNO provides an error identifier if an invalid value has been applied to an input or if an error occurred during job processing. ERNO always has to be considered together with the outputs DONE and ERR. The output value at ERNO is only valid if DONE = TRUE and ERR = TRUE.

Call in ST

```
PM5xxSend (EN := PM5xxSend_EN,
DATA := ADR(PM5xxSend_DATA),
DATA_LEN := PM5xxSend_DATA_LEN,
DONE => PM5xxSend_DONE,
ERR => PM5xxSend_ERR,
ERNO => PM5xxSend_ERNO);
```

5 Safety times

5.1 Overview

Errors in the system may lead to dangerous operating conditions. Potential errors are detected by the safety module background self-tests, which trigger defined error reactions in safety modules to transfer faulty modules into the safe state. In this chapter, we list various safety times for AC500-S safety modules and AC500-S safety PLC as a system.

5.2 Fault reaction time

Fault reaction time is the maximum time between the appearance of the fault in the system and the trigger of pre-defined error reactions. The table below provides an overview on the longest fault reaction times in AC500-S safety modules.

Table 92: Fault reaction times in AC500-S safety modules

Module	**Fault reaction time**	
	Internal faults (e.g., RAM cell fault)	**External faults (e.g., wrong wiring)**
AC500-S safety CPUs	< 24 h	Not applicable
DI581-S safety I/O	< 24 h	< 1.9 s
DX581-S safety I/O	< 24 h	< 0.5 s
AI581-S safety I/O	< 24 h	< 0.8 s

Contact ABB technical support for more detailed fault reaction times, if needed.

5.3 Safety function response time

The safety function response time (SFRT) is the time within which the AC500-S safety PLC in the normal RUN mode must react after an error has occurred in the system.

On the application side, SFRT is the maximum amount of time in which the safety system must respond to a change in input signals or module failures.

SFRT is one of the most important safety times, because it is used in time-critical safety applications, like presses, to define a proper distance for a light curtain or other safety sensor to protect people from potentially dangerous machine parts.

SFRT for PROFIsafe devices can be defined as, based on ↳ *[8]*:

Equation 1: *SFRT = TWCDT + Longest ∆T_WD*

where

- TWCDT (total worst case delay time) is the maximal time for input signal transfer in AC500-S system until the output reaction under worst-case conditions (all components require the maximum time).
- Longest ∆T_WD is the longest time difference between watchdog time for a given entity and worst case delay time. In safety context, to identify SFRT one has to take into account a potential single fault in one of the safety modules during the signal transfer. It is enough to consider a single fault only ↳ *[8]*.

Fig. 122, Fig. 123 and Fig. 124 explain SFRT in more details. The model in Fig. 122 and Fig. 123 includes the stages of input signal reading, safe data transfer, safe logic processing, safe data transfer and safe signal output. The model in Fig. 124 presents safe CPU to CPU communication, which includes the stages of safe logic processing, safe data transfer and safe logic processing.

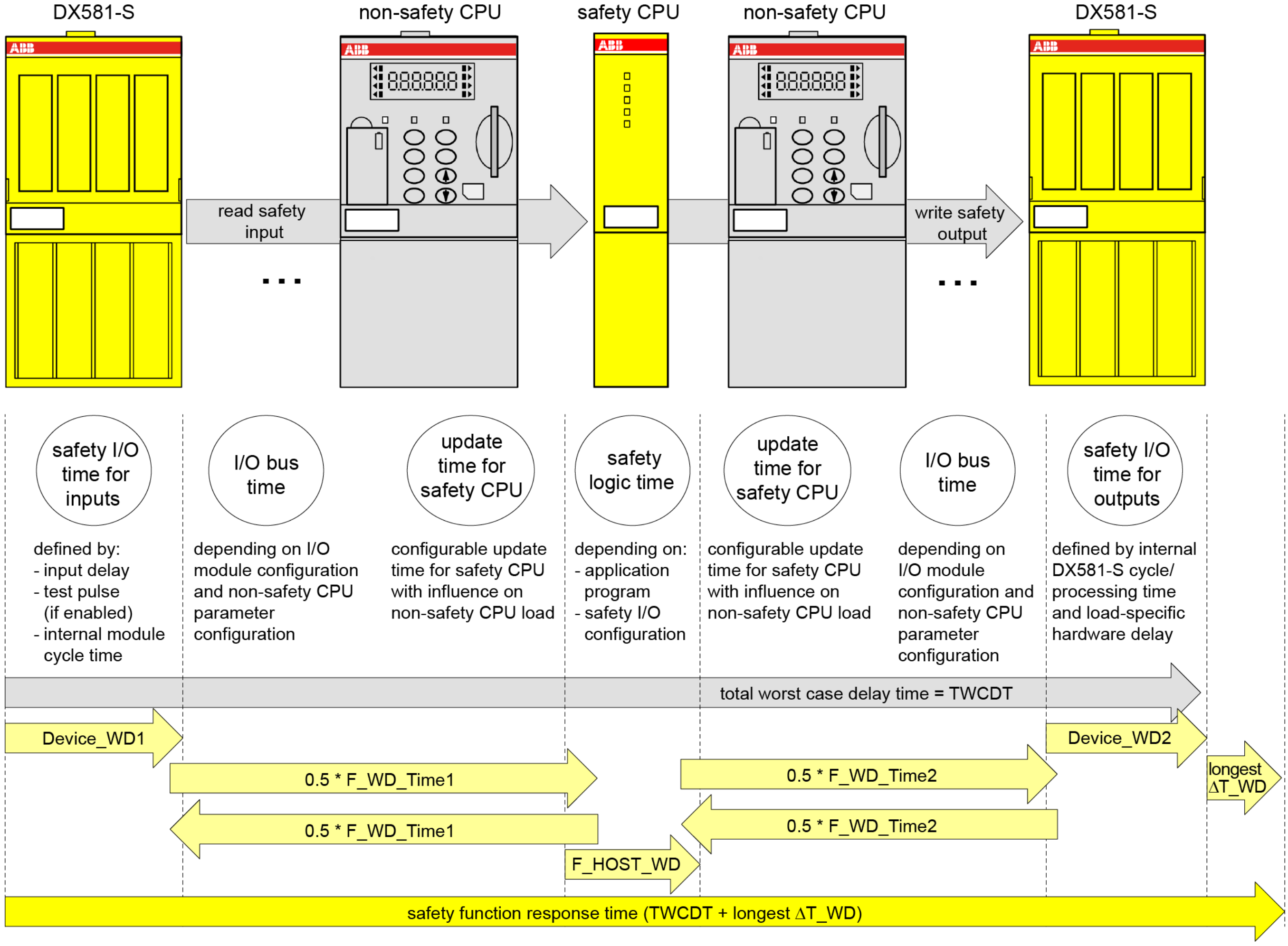


Fig. 122: SFRT in AC500-S system without PROFINET components

All terms in this figure are further explained ⇘ *on page 328*.

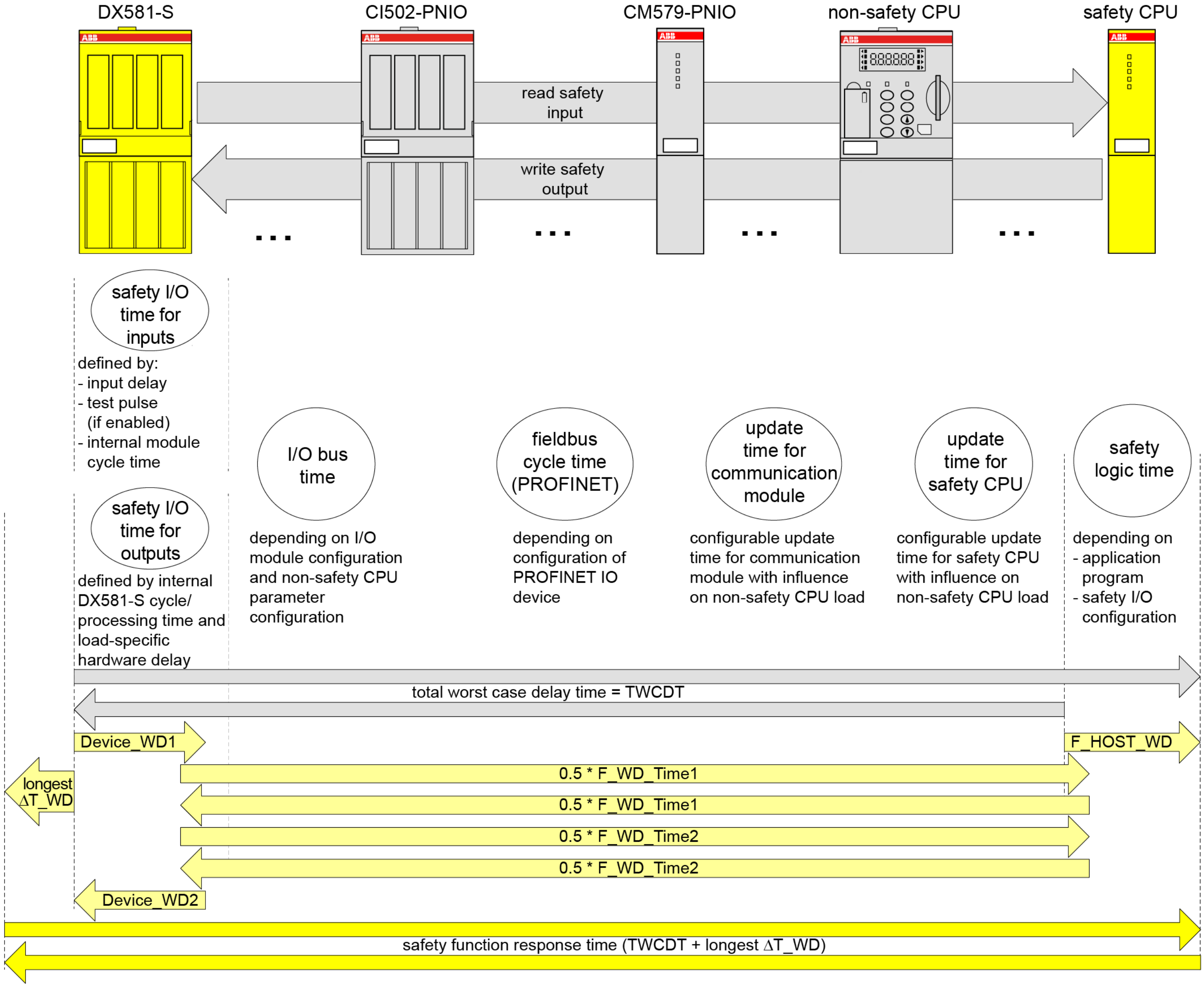


Fig. 123: SFRT in AC500-S system with PROFINET components and safety I/O modules

All terms in this figure are further explained *on page 328*.

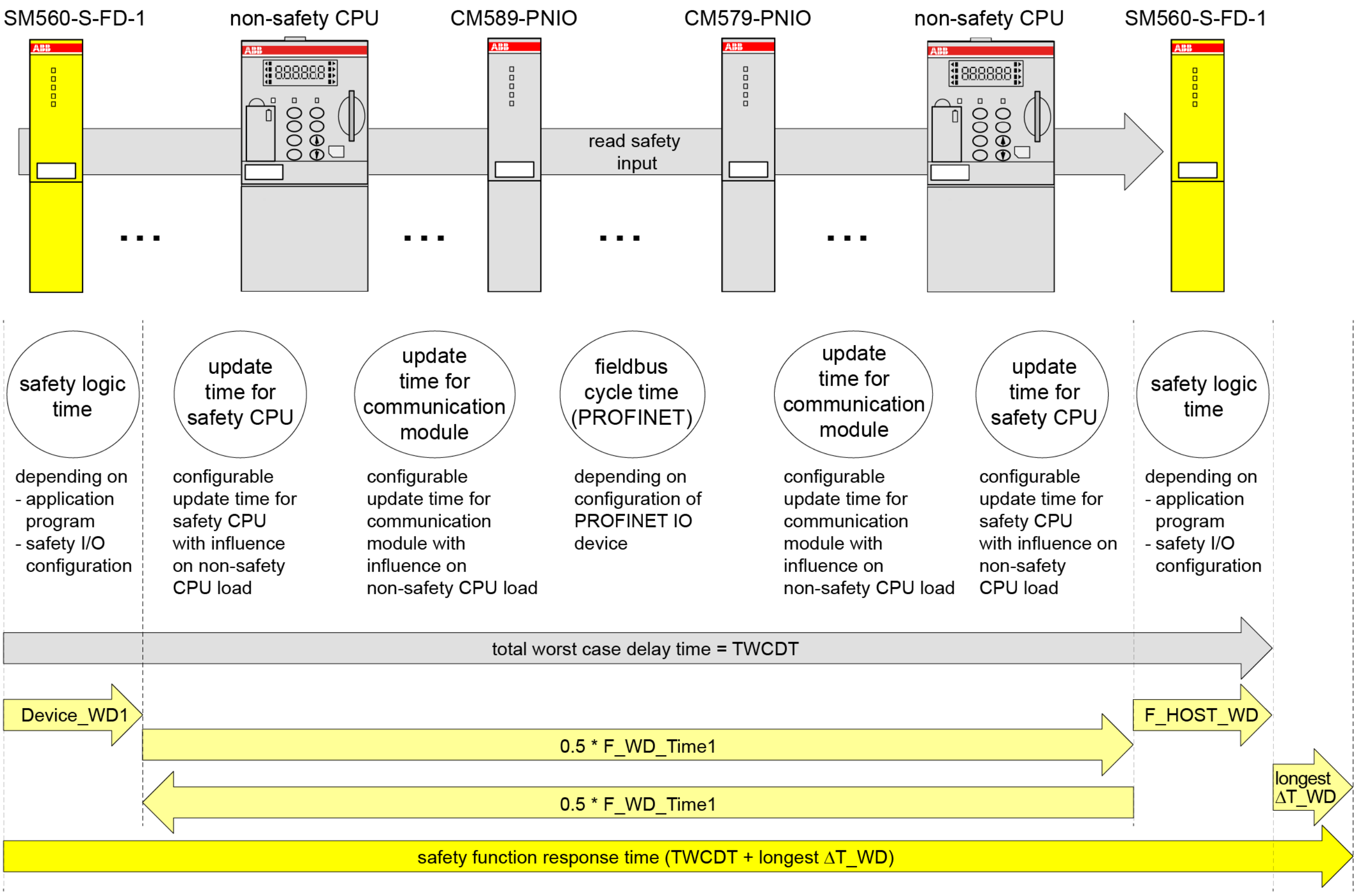


Fig. 124: SFRT in AC500-S system with PROFINET components and safe CPU to CPU communication (example: SM560-S-FD-1 to SM560-S)

All terms in this figure are further explained *on page 328*.

Explanation of terms related to SFRT

The following terms are defined in Fig. 122, Fig. 123 and Fig. 124 (in alphabetical order):

- **Device_WD1 (safety I/O time for inputs)** is an internal input device watchdog time in ms which includes:
  – Input delay (variable as parameter; not used for safety analog inputs which have an internal input delay of 67.5 ms in the worst case instead).
  – Input delay accuracy *Table 4 "Input delay accuracy for DI581-S" on page 65* *Table 6 "Input delay accuracy for DX581-S" on page 90*.
  – Test pulse low phase (fixed to 1 ms and optional (only if test pulses are used); not used for safety analog inputs).
  – Two times internal cycle time (fixed; AI581-S ➔ 4.5 ms, DX581-S ➔ 5.5 ms and DI581-S ➔ 6.5 ms).
- **Device_WD2 (safety I/O time for outputs)** is an internal output device watchdog time in ms which includes:
  – Internal safety output device cycle time (fixed; DX581-S ➔ 5.5 ms).
  – Output processing time in DX581-S (fixed to 1.5 ms).
  – Hardware delay (current dependent, e.g., ~1 ms (747 µs at 5 mA) and the maximum of 4 ms under the maximum output current of 500 mA). If more precise values are needed, please contact ABB technical support.
- **F_Host_WD (safety logic time)** is the time which can be calculated as three times safety application cycle watchdog time. The safety application cycle watchdog time itself is configurable using POU SF_WDOG_TIME_SET. The safety application cycle watchdog time depends on the number of F-Devices, safety application program and system configuration.

- **F_WD_Time1** and **F_WD_Time2**: The sum represents the data transport time in total via „black channel". It covers different "black channel" components, like fieldbus cycle time (PROFINET), I/O bus time and update time for safety CPU (configurable as parameter) and communication module.
- **Fieldbus cycle time (PROFINET)** depends on the communication settings for the PROFINET IO device where the safety I/O module is attached to. The cycle time is a multiplication of two parameters of the PROFINET IO device.
  – *"Send clock"*, e.g., for CI501-PNIO and CI502-PNIO: 1 ms, 2 ms or 4 ms
  – *"Reduction ratio"*, e.g., for CI501-PNIO and CI502-PNIO: 1, 2, 4, 8, 16 ... 512

  These values can be selected depending on the defined PROFINET parameters for this PROFINET module.
- The configurable **update time for safety CPU and communication modules** describes the data transfer time via the communication module bus.
  – With AC500 V2 non-safety CPU:
    The update time can be configured within the range of 0 ... 20000 ms for both safety CPU and communication modules.
  – With AC500 V3 non-safety CPU:
    The update time for safety CPU can be configured within the range of 1 ... 20000 ms.
    The update time for communication modules is related to PROFINET IO controller (CM579-PNIO) and PROFINET IO device (CM589-PNIO) settings. It is defined by the communication module setting *"Bus cycle task"*, e.g., in tab *"PROFINET-IO-Controller I/O Mapping"*. Additional information: *""Bus cycle task"" on page 398*
- **I/O bus time** describes the data transfer time via I/O bus for communication between non-safety CPU and its local I/O bus modules as well as for communication between communication interface modules and their local I/O bus modules.
  – With AC500 V2 non-safety CPU:
    The I/O bus cycle time has no fixed pre-defined cycle value. It is defined by the number and type of the configured I/O modules independent from non-safety CPU settings. The I/O bus time contains the following values:
    - I/O bus master cycle: 2 ms (2 cycles, 1 ms each)
    - I/O bus cycle time: Typically 2 ... 5 ms (2 cycles, 1 ... 2.5 ms each)
    In total, the typical range for the I/O bus time is 4 ... 7 ms.
  – With AC500 V3 non-safety CPU:
    The I/O bus is driven with a defined cycle time. This I/O bus cycle time relates to non-safety CPU setting *"Bus cycle task"* in tab *"I/O-Bus I/O Mapping"*. Refer to additional information: *""Bus cycle task"" on page 398*.
    A basic definition of I/O bus cycle times is done for non-safety CPU in setting *"Bus cycle task"* in tab *"PLC settings"*.
    Example for a setting with assignment to a task with 2 ms cycle time (and lower than the defined update time for safety CPU):
    - Result for I/O bus master cycle: 2 ms = 2 cycles, 1 ms each
    - Result for I/O bus cycle time: Typically 4 ... 5 ms = 2 cycles, 2 ... 2.5 ms each (if the configured task cycle time does not fit to the I/O bus module assembly, the I/O bus cycle time can be extended to a maximum of 2.5 ms)
    In total, the I/O bus time for this example is 6 ... 7 ms.
    Refer to additional information: *""Bus cycle task"" on page 398*, e.g., for I/O bus.

Below, a few examples on how to calculate SFRT values under various AC500-S system configurations are presented. In our calculations, we use the following approach, based on *[3]* and *[8]*, which allows us calculating SFRT as:

Equation 2: *SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + 0.5 * F_WD_Time2 + Device_WD2 + Longest ∆T_WD*

DANGER!
Input delay, input delay accuracy and test pulse low phase are not needed for AI581-S. However, the worst case fixed internal input delay of 67.5 ms shall be used for AI581-S instead.

DANGER!
The input delay accuracy has to be calculated based on the following assumptions:

- It is not used for safety analog inputs.
- If no test pulses are configured for the given safety digital input, then input delay accuracy can be calculated as 1 % of set input delay value (however, input delay accuracy value must be at least 0.5 ms!).
- If test pulses are configured for the given safety digital input, then depending on the type of the module (DI581-S or DX581-S) and set input delay value, the following input delay accuracy values can be used in SFRT calculations: ↳ *Table 4 “Input delay accuracy for DI581-S” on page 65* ↳ *Table 6 “Input delay accuracy for DX581-S” on page 90*

NOTICE!
↳ *Equation 2 on page 329* is taken for SFRT calculation with the following reasoning:

- Device_WD1 and Device_WD2, as worst case delay times for safety I/Os, can be defined as it is shown in Fig. 122 and Fig. 123.
- To calculate the worst case delay time of "Black channel" components (refer to AC500 non-safety modules in Fig. 122 and Fig. 123), we propose to use half of F_WD_Time1 and F_WD_Time2 instead. F_WD_Time1 and F_WD_Time2 can be empirically obtained for the given AC500 system configuration by tracing the values of tResponseTimeMS for given safety I/Os in the safety application. Use PROFIsafe instance for the given safety I/O ↳ *Chapter 4.6.3 “SafetyBase_PROFIsafe_LV200_AC500_V22.lib” on page 194*. F_WD_Time1 and F_WD_Time2 shall be set about 30 % higher than the worst case value observed in the tResponseTimeMS for the given safety I/O.
- We propose to take F_Host_WD time instead of the worst case delay time of SM560-S safety CPU. F_Host_WD can be calculated as three times the value set using SF_WDOG_TIME_SET POUs. The correct value for SF_WDOG_TIME_SET can be empirically obtained using tracing MAX_TIME output of the same POU in a test run. SF_WDOG_TIME_SET value shall be set about 30 % higher than the worst case value (MAX_TIME) observed in the given safety application to avoid potential availability problems due to triggering of SM560-S safety CPU watchdog.
- F_WD_Time1 and F_WD_Time2 are the only potential candidates for longest ΔT_WD, because F_Host_WD, Device_WD1 and Device_WD2 are already equal to their worst case delay times. Thus,

Longest ΔT_WD = Max (0.5 * F_WD_Time1; 0.5 * F_WD_Time2)

NOTICE!
One could achieve even better SFRT values than those obtained using ↳ *Equation 2 on page 329* with a more detailed technical analysis. Contact ABB technical support for further details.

NOTICE!

You have to set F_WD_Time1 and F_WD_Time2 at least 2 times bigger than the value set using SF_WDOG_TIME_SET time to avoid unintended system stop due to PROFIsafe watchdog expiration.

DANGER!

AC500-S safety I/O modules satisfy the requirement of IEC 61131 to bypass a potential undervoltage event with a duration of up to 10 ms. During this under-voltage effect of up to 10 ms, AC500-S safety I/O modules deliver the last valid process value before the undervoltage was detected for safety analog input channels in AI581-S and actual safety digital input and output values for DI581-S and DX581-S modules.

If the undervoltage phase is longer than 10 ms then safety I/O module passivation occurs *Chapter 3.2.3 “Undervoltage / overvoltage” on page 61*.

If undervoltage events with duration of < 10 ms are frequently observed in the safety application, you have to add 10 ms for AI581-S module in their SFRT calculation to take into account a bypass stage described above. Normally, seldom undervoltage events with duration of < 10 ms are considered to be low probability faults in the power supply of the safety system and can be omitted in the SFRT calculation.

Based on Fig. 122, Fig. 123 and Fig. 124, the following exemplary SFRT values can be achieved for some typical AC500-S configurations using *Equation 2 on page 329*:

Without PROFINET (DI581-S ➔SM560-S ➔ DX581-S)

SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + 0.5 * F_WD_Time2 + Device_WD2 + Longest ΔT_WD = 14.5 + 10 + 6 + 10 + 8 + 10 = 58.5 ms

where:

- Device_WD1 = 1 ms + 0.5 ms + 2 x 6.5 ms = 14.5 ms (no test pulses were used)
- F_WD_Time1 = 20 ms
- F_Host_WD = 3 x 2 ms (SF_WDOG_TIME_SET time) = 6 ms
- F_WD_Time2 = 20 ms
- Device_WD2 = 8 ms (output current = ~ 5 mA)
- Longest ΔT_WD = Max (0.5 * F_WD_Time1; 0.5 * F_WD_Time2) = 10 ms

Without PROFINET (DX581-S ➔ SM560-S ➔ DX581-S)

SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + 0.5 * F_WD_Time2 + Device_WD2 + Longest ΔT_WD = 12.5 + 10 + 6 + 10 + 8 + 10 = 56.5 ms

where:

- Device_WD1 = 1 ms + 0.5 ms + 2 x 5.5 ms = 12.5 ms (no test pulses were used)
- F_WD_Time1 = 20 ms
- F_Host_WD = 3 x 2 ms (SF_WDOG_TIME_SET time) = 6 ms
- F_WD_Time2 = 20 ms
- Device_WD2 = 8 ms (output current = ~ 5 mA)
- Longest ΔT_WD = Max (0.5 * F_WD_Time1; 0.5 * F_WD_Time2) = 10 ms

Without PROFINET (AI581-S ➔ SM560-S ➔ DX581-S)

SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + 0.5 * F_WD_Time2 + Device_WD2 + Longest ∆T_WD = 76.5 + 10 + 6 + 10 + 8 +10 = 120.5 ms

where:

- Device_WD1 = 2 x 4.5 ms + 67.5 ms = 76.5 ms
- F_WD_Time1 = 20 ms
- F_Host_WD = 3 x 2 ms (SF_WDOG_TIME_SET time) = 6 ms
- F_WD_Time2 = 20 ms
- Device_WD2 = 8 ms (output current = ~ 5 mA)
- Longest ∆T_WD = Max (0.5 * F_WD_Time1; 0.5 * F_WD_Time2) = 10 ms

With PROFINET (DI581-S ➔ SM560-S ➔ DX581-S)

SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + 0.5 * F_WD_Time2 + Device_WD2 + Longest ∆T_WD = 14.5 + 15 + 6 + 15 + 8 + 15 = 73.5 ms

where:

- Device_WD1 = 1 ms + 0.5 ms + 2 x 6.5 ms = 14.5 ms (no test pulses were used)
- F_WD_Time1 = 30 ms
- F_Host_WD = 3 x 2 ms (SF_WDOG_TIME_SET time) = 6 ms
- F_WD_Time2 = 30 ms
- Device_WD2 = 8 ms (output current = ~ 5 mA)
- Longest ∆T_WD = Max (0.5 * F_WD_Time1; 0.5 * F_WD_Time2) = 15 ms

With PROFINET (DX581-S ➔ SM560-S ➔ DX581-S)

SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + 0.5 * F_WD_Time2 + Device_WD2 + Longest ∆T_WD = 12.5 + 15 + 6 + 15 + 8 + 15 = 71.5 ms

where:

- Device_WD1 = 1 ms + 0.5 ms + 2 x 5.5 ms = 12.5 ms (no test pulses were used)
- F_WD_Time1 = 30 ms
- F_Host_WD = 3 x 2 ms (SF_WDOG_TIME_SET time) = 6 ms
- F_WD_Time2 = 30 ms
- Device_WD2 = 8 ms (output current = ~ 5 mA)
- Longest ∆T_WD = (Max (0.5 * F_WD_Time1; 0.5 * F_WD_Time2) = 15 ms

With PROFINET (AI581-S ➔ SM560-S ➔ DX581-S)

SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + 0.5 * F_WD_Time2 + Device_WD2 + Longest ∆T_WD = 76.5 + 15 + 6 + 15 + 8 + 15 = 135.5 ms

where:

- Device_WD1 = 2 x 4.5 ms + 67.5 ms = 76.5 ms
- F_WD_Time1 = 30 ms
- F_Host_WD = 3 x 2 ms (SF_WDOG_TIME_SET time) = 6 ms
- F_WD_Time2 = 30 ms
- Device_WD2 = 8 ms (output current = ~ 5 mA)
- Longest ∆T_WD = Max (0.5 * F_WD_Time1; 0.5 * F_WD_Time2) = 15 ms

With PROFINET (SM560-S-FD-1 ➔ SM560-S)

SFRT = Device_WD1 + 0.5 * F_WD_Time1 + F_Host_WD + Longest ∆T_WD = 9 + 25 + 6 + 25 = 65 ms

where:

- Device_WD1 = 3 x 3 ms (SF_WDOG_TIME_SET time) = 9 ms
- F_WD_Time1 = 50 ms
- F_Host_WD = 3 x 2 ms (SF_WDOG_TIME_SET time) = 6 ms
- Longest ∆T_WD = 0.5 * F_WD_Time1 = 25 ms

NOTICE!
SFRT calculation for such cases as SM560-S-FD-4 ➔ SM560-S, SM560-S ➔ SM560-S-FD-1, SM560-S ➔ SM560-S-FD-4, etc. can be calculated in a similar way as it is shown in Fig. 124.

DANGER!
Mistakes in SFRT calculation can lead to death or severe personal injury, especially in such applications like presses, robotic cells, etc.

NOTICE!
The high priority tasks on non-safety CPU, which are a part of the "black channel" for safety communication, may affect TWCDT for AC500-S safety PLC.

6 Checklists for AC500-S commissioning

6.1 Overview

All users of AC500-S safety PLC shall evaluate items from the checklists presented in this chapter for AC500-S commissioning and document those in their final reports.

The items presented in the checklists include only the most important ones from AC500-S safety PLC perspective, which means that AC500-S checklists can be also extended by users to include additional aspects important for their safety applications.

6.2 Checklist for creation of safety application program

No.	Item to check	Fulfilled (yes / no)?	Comment
1.	Verify that only safety signals are used for all safety functions.		
2.	Verify that not only safety application project is loaded to the safety CPU but also the relevant non-safety application project is loaded to non-safety CPU. Verify that programs are saved from RAM memory to the flash memory, i.e., "Create boot project" is done.		
3.	Up to Automation Builder 2.2.x: Verify that F-Parameters for all safety I/Os and other F-Devices set in F-Parameter editor are the same as those listed in CODESYS Safety: *“Global Variables ➔ PROFIsafe”* *Chapter 4 “Configuration and programming” on page 133*. Automation Builder 2.3.x (and higher): Verify that a valid SVT report is present for the application project.		
4.	F-Host on safety CPU can handle more than one F_Source_Add, if required, e.g., for PROFIsafe master - master coupling of different network islands. Verify that no ambiguous F_Source_Add settings for various F-Devices were set for the given safety application. *Note:* *The rule "F_Source_Add <> F_Dest_Add for the given F-Device" is automatically checked by Automation Builder.*		
5.	Validate iParameters. Two options are available: A) Validate that all iParameters (input delay, channel configuration, etc.) for all safety I/Os and other F-Devices are correct with a given F_iPar_CRC value using appropriate functional validation tests for those parameters (contact ABB technical support for more details) or B) Use a special verification procedure defined in *Chapter 6.5 “Verification procedure for safe iParameter setting in AC500-S safety I/Os” on page 341* to validate each iParameter and then carry out only functional safety validation tests of your application (no need to check each single iParameter value). You have to provide a report confirming that all iParameters were checked as described in *Chapter 6.5 “Verification procedure for safe iParameter setting in AC500-S safety I/Os” on page 341*. Make sure that all F_iPar_CRC are > 0.		

No.	Item to check	Fulfilled (yes / no)?	Comment
6.	Verify that CODESYS Safety programming guidelines were properly used in the safety application program *Chapter 4.4 "CODESYS Safety programming guidelines" on page 179*.		
7.	All signals from the non-safety user program on non-safety CPU, which are evaluated in the safety program on the safety CPU, have to be also included when the safety application program is printed out.		
8.	Has a review of the safety application program been carried out by a person not involved in the program creation?		
9.	Has the result of the safety application program review been documented and released (date/ signature)?		
10.	Was a backup of the complete safety (see note below) and non-safety project created before loading programs on safety and non-safety CPUs? *Note:* ● *Make sure that file name, change date, title, author, version, description and CRC of the CODESYS Safety boot project are documented as a backup.* ● *No further changes are allowed for safety parts in Automation Builder project and CODESYS Safety. If any changes are still done, then they will lead to a new CODESYS Safety boot project CRC, which will require re-doing this checklist from the beginning.*		
11.	Verify using CODESYS Safety menu item *"Online ➔ Check boot project in PLC"* that offline CODESYS Safety project and the boot project on the safety CPU are identical (file name, change date, title, author, version, description and CRC).		
12.	If floating-point operations are used, verify that rules presented in *Chapter 3.1.2.2 "Floating-point operations" on page 34* are taken into account and do not lead to any unsafe states in the safety application program.		
13.	Verify that POU SF_WDOG_TIME_SET is called once in the safety application program and the watchdog time is correctly selected.		
14.	Verify that a password for the safety CPU is set to prevent an unauthorized access to its data.		
15.	Verify that only authorized personnel has *"Write"* access for safety module parameter settings and programs in Automation Builder and CODESYS Safety projects.		
16.	Verify that correct value for power supply supervision using POU SF_MAX_POWER_DIP_SET was set to have a correct system behavior in case of under- or over-voltage.		
17.	Verify that POU SF_SAFETY_MODE is correctly used in the safety application program to avoid unintended safety program execution in DEBUG (non-safety) mode.		

No.	Item to check	Fulfilled (yes / no)?	Comment
18.	Verify that no profile version change, *"Update Device..."*, Export/Import, Copy/Paste and Archive related functions in Automation Builder were executed on safety modules after the project was validated. If the functions mentioned above were used and this leads to a CODESYS Safety boot project with a new CRC, then a full functional testing of all parts of the safety application has to be performed. This test must be carried out with the machine in its final configuration including mechanical, electrical and electronic components, sensors, actuators, and software.		
19.	Verify using library CRC, shown in CODESYS Safety, that only certified safety libraries with correct CRCs (refer to *Chapter 4.6.1 "Overview" on page 189*) are used in the given CODESYS Safety project to execute safety functions. All other user-defined libraries have to be separately validated by the end-user to qualify for the given safety application.		
20.	Make sure that internal POUs from SafetyUtil_CoDeSys_AC500_V22.lib and internal actions from SafetyBase_PROFIsafe_LV200_AC500_V22.lib (or older versions) are not called by end-user program, which starts from PLC_PRG as the main root.		
21.	Make sure that all three system events (*"CallbackInit"*, *"CallbackReadInputs"* and *"CallbackWriteOutputs"*) in *"Resources ➔ Task configuration ➔ System Events"* of CODESYS Safety remain selected.		
22.	If the flash memory content (SF_FLASH_READ and/or SF_FLASH_WRITE FBs are called in the safety application) is used in the safety application for safety functions, then appropriate flash memory content validation procedures (e.g., proper safety application CRC over stored safety data) shall be implemented to ensure safety application data integrity before flash memory data are used in safety functions.		
23.	Verify ● that the symbolic variables of configured F-Devices are mapped properly and ● that the delivered safety data is correctly represented in your safety application. I.e., if data types which require more than one byte (like Unsigned16, Unsigned32, Integer16, Integer32, Float32) are used in PROFIsafe data. *Note:* *The byte order in PROFIsafe data types depends on the used PROFIsafe device endianness and selected AC500 CPU type. (AC500 V2 non-safety CPU supports big-endian. AC500 V3 non-safety CPU supports little-endian.)*		

No.	Item to check	Fulfilled (yes / no)?	Comment
24.	If you use cyclic non-safe data exchange, make sure that only safety functions with up to SIL 2 (IEC 61508 and IEC 62061) and PL d (ISO 13849-1) will be triggered if sending data using cyclic non-safe data exchange. *Note:* *If cyclic non-safe data exchange is used to send or receive safety-critical data, then SIL 3 (IEC 61508 and IEC 62061) and PL e (ISO 13849-1) safety requirements will not be fulfilled for sent or received data (independently on application safety communication profile used), because only one microprocessor (no 1oo2 safety architecture in the background) on safety CPU handles the sending and receiving direction.* *Contact ABB technical support on how to reach SIL 3 and PL e.*		
25.	If you use cyclic non-safe data exchange, verify that the variable names of cyclic non-safe data exchange which are created for the safety CPU do not start with "S_", "GS_", "IS_" or "OS_".		
Reviewer(s): Machine/Application <ID>: Signature: Date:			

6.3 Checklist for configuration and wiring

No.	Item to check	Fulfilled (yes / no)?	Comment
1.	Are all safety input and output signals correctly configured and are the output signals connected to physical output channels?		
2.	Verify that safety CPU switch addresses 0xF0 ... 0xFF are not used for safety CPU identification (e.g., PROFIsafe addresses).		
3.	Verify that special organizational procedures (e.g., limited access to the cabinet where safety CPU is located) on the end-customer site are defined to avoid unintended firmware and/or boot code update on the safety CPU using SD card.		
4.	Verify that correct parameter settings of non-safety CPU are used for the given safety application *Appendix B.3 “AC500 V2 non-safety CPU parameters configuration” on page 382* *Appendix C.3 “AC500 V3 non-safety CPU parameters configuration” on page 397*.		
5.	Verify that required safety function response time of your safety application can be satisfied with current AC500-S safety PLC settings and your SFRT calculation is done based on *Chapter 5.3 “Safety function response time” on page 325*.		

No.	Item to check	Fulfilled (yes / no)?	Comment
6.	Verify that none of safety output channels has a configuration with "Detection" parameter = OFF, which reduces safety diagnostics for such safety output channels. If such configuration is used, explain in the "Comment" section of this checklist your reasons and claim that the required SIL and PL application levels can be reached with such configuration.		
7.	Verify that: ● Address setting is correct. ● Assignment of signal inputs is complete. ● Assignment of signal outputs is complete. ● Assignment of unused inputs is complete. ● All terminal blocks are plugged.		
8.	Verify that correct firmware versions are used for dependent non-safety components *Appendix B.1 “Compatibility with AC500 V2 non-safety CPU” on page 373* *Appendix C.1 “Compatibility with AC500 V3 non-safety CPUs” on page 391*. Contact ABB technical support if needed.		
9.	Verify that only one safety CPU is attached to non-safety CPU. The use of more than one safety CPU on one non-safety CPU is not allowed.		
10.	Verify that the correct CODESYS Safety boot project is loaded on the right AC500-S safety CPU, for example, using organizational procedures or fault exclusion (only one safety CPU is available in the machine). Examples of organizational procedures are: ● If engineering PC is used and there is more than one safety CPU, then make sure that only one and right safety CPU is reachable for engineering PC when the given CODESYS Safety boot project is transferred to the safety CPU. ● If SD card is used and there is more than one safety CPU, then clearly identify each safety CPU and SD card using a proper ID marking on stickers attached to each safety CPU and SD card. These ID markings on stickers shall provide a clearly readable unique identification of each object to establish clear rules for relations "SD card with given CODESYS Safety boot project - Safety CPU".		
11.	Verify that the following rules were correctly applied for safe CPU to CPU communication using SM560-S-FD-1 and SM560-S-FD-4 CPUs: ● In the same codename space, F_Dest_Add shall be unique (Fig. 6 on page 39). ● In the same codename space, F_Source_Add shall not be re-used in other F-Hosts. Inside the same F-Host, a re-use is allowed for several F-Host Drivers. ● In the same codename space, F_Dest_Add shall not be used as F_Source_Add and vice versa.		
12.	If SM560-S-FD-1 or SM560-S-FD-4 is used, make sure that safety communication modules ("12 Byte In/Out (Safety)" / "8 Byte and 2 Int In/Out (Safety)" are correctly connected to master systems.		

No.	Item to check	Fulfilled (yes / no)?	Comment
13.	Verify that not only codenames but also F_Dest_Add are unique in PROFIsafe networks, if only F_Dest_Add is checked by the F-Device.		
Reviewer(s): Machine/Application <ID>: Signature: Date:			

6.4 Checklist for operation, maintenance and repair

No.	Item to check	Fulfilled (yes / no)?	Comment
1.	Make sure that all safety modules are properly placed on their positions at the terminal base for safety CPU or terminal units for safety I/Os and stable contact between terminals and safety modules is assured.		
2	Check that proper temperature monitoring measures (e.g., temperature sensors could be placed in the control cabinet and connected to AI581-S safety analog input channels) are implemented in the control cabinet where AC500-S safety modules are placed, if the operating temperature range for AC500-S safety PLC cannot be guaranteed. *Note:* *Safety digital outputs of DX581-S module have internal built-in overtemperature protection and always deliver fail-safe "0" values in case of overtemperature.*		
3.	Make sure that the following rule, as defined by PROFIsafe standard (refer to *www.profisafe.net* for more details), was considered in the safety application analysis: ● A maximum of 10 communication links (i.e., PROFIsafe connections from the given safety input to the given safety output) per safety function is permitted for an average probability of a dangerous failure of 10^{-9}/h (SIL 3). In case of more than 10 communication links per safety function, the probability of a dangerous failure increases by 10^{-10}/h per additional communication link. Correspondingly, a maximum of 100 communication links is permitted in case of SIL 2.		
4.	Make sure that all network devices used in conjunction with AC500-S safety PLC meet the requirements of IEC 61010 or IEC 61131-2 (e.g., PELV). Single port routers are not permitted as borders for a safety island. Refer to ⇘ *[3]* for further details.		

No.	Item to check	Fulfilled (yes / no)?	Comment
5	Before any deployment of a safety application with PROFIsafe, especially those using wireless components, an assessment for dangerous threats such as eavesdropping or data manipulation shall be executed (refer to ➥ *[11]* for more details). Check that adequate level of security defining security zones with security gates was established. In case of no threat, no security measures are necessary. *Note:* *There are two possible threats identified so far mainly for applications with wireless components ➥ [3]:* ● *Willful changes of parameters of F-Devices and safety programs.* ● *Attacks on the cyclic communication, e.g., simulation of the safety communication.*		
6.	The complete functional testing of all parts of the safety application has to be performed. This test must be carried out with the machine in its final configuration including mechanical, electrical and electronic components, sensors, actuators, and software.		
7.	Verify that clear operation, maintenance and repair procedures (organization, responsibility, spare parts, project data backup, etc.) for safety application are defined. Note: ● Restart of the corresponding safety control loop is only permitted, if there is no hazardous process state, and after an operator acknowledgment (OA_C). Refer to ➥ *[3]* for further details.		
8.	Verify that proper electrical contact is available between safety I/O modules (AI581-S, DI581-S and DX581-S) and TU582-S terminal units. Follow the assembly instructions for safety I/O modules ➥ *“Assembly of DI581-S” on page 68* ➥ *“Assembly of DX581-S” on page 94* ➥ *“Assembly of AI581-S” on page 113*.		
9.	Ensure that average operating temperature for used safety modules (AC500-S and AC500-S-XC) does not exceed +40 °C (e.g., temperature sensors could be placed in the control cabinet and connected to AI581-S safety analog input channels for temperature monitoring).		
10.	Verify that no automatic reboot of non-safety CPU is programmed in CODESYS non-safety program. The automatic reboot of non-safety CPU would lead to automatic restart of the safety CPU, which is directly attached to non-safety CPU. Such automatic restart of the safety CPU may not be accepted in some safety applications.		
Reviewer(s): Machine/Application <ID>: Signature: Date:			

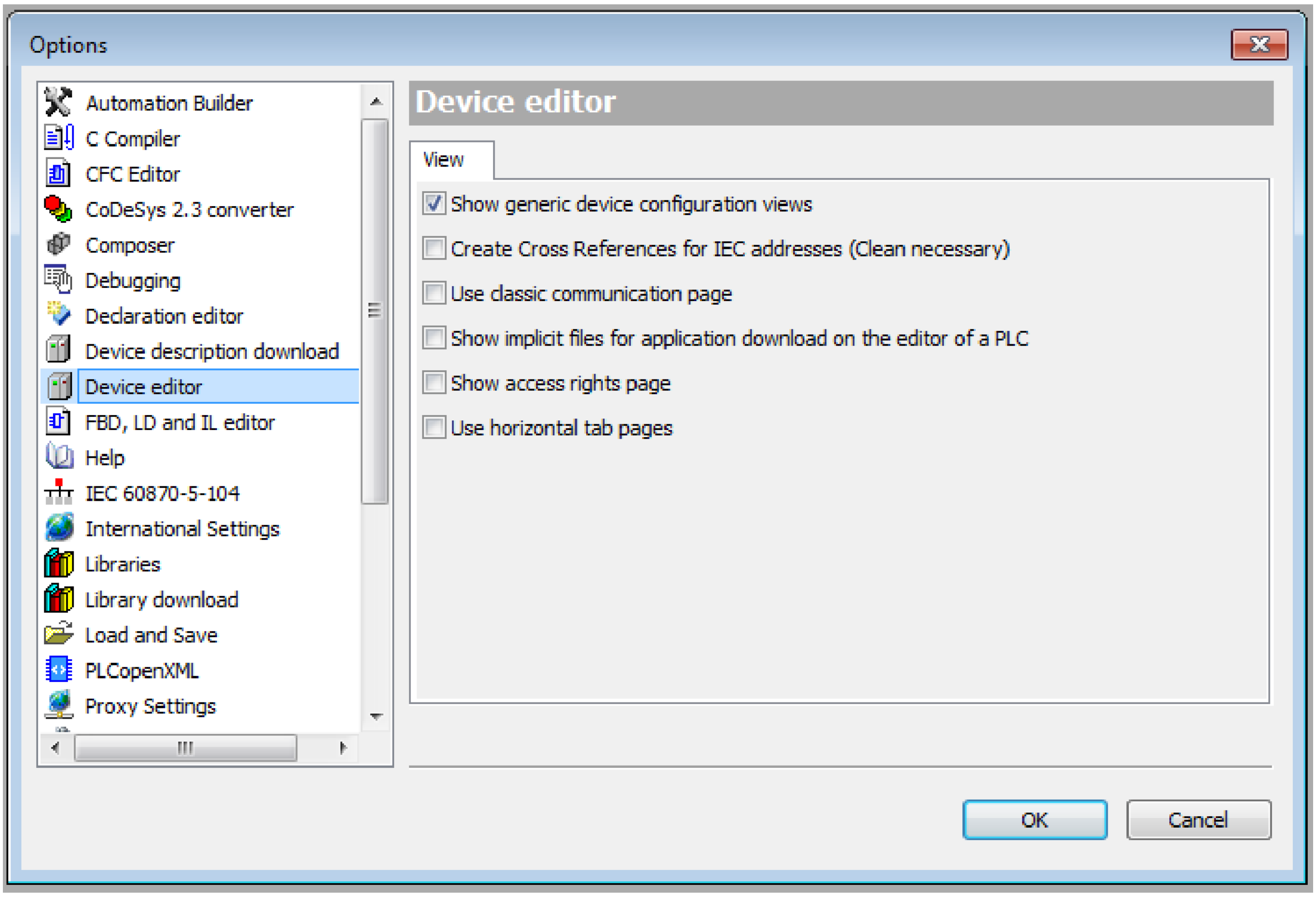
6.5 Verification procedure for safe iParameter setting in AC500-S safety I/Os

This verification procedure has to be performed before commissioning of the final safety application and relevant validation tests to confirm that F_iPar_CRC was calculated for a correct set of iParameters.

6.5.1 Verification procedure workflow

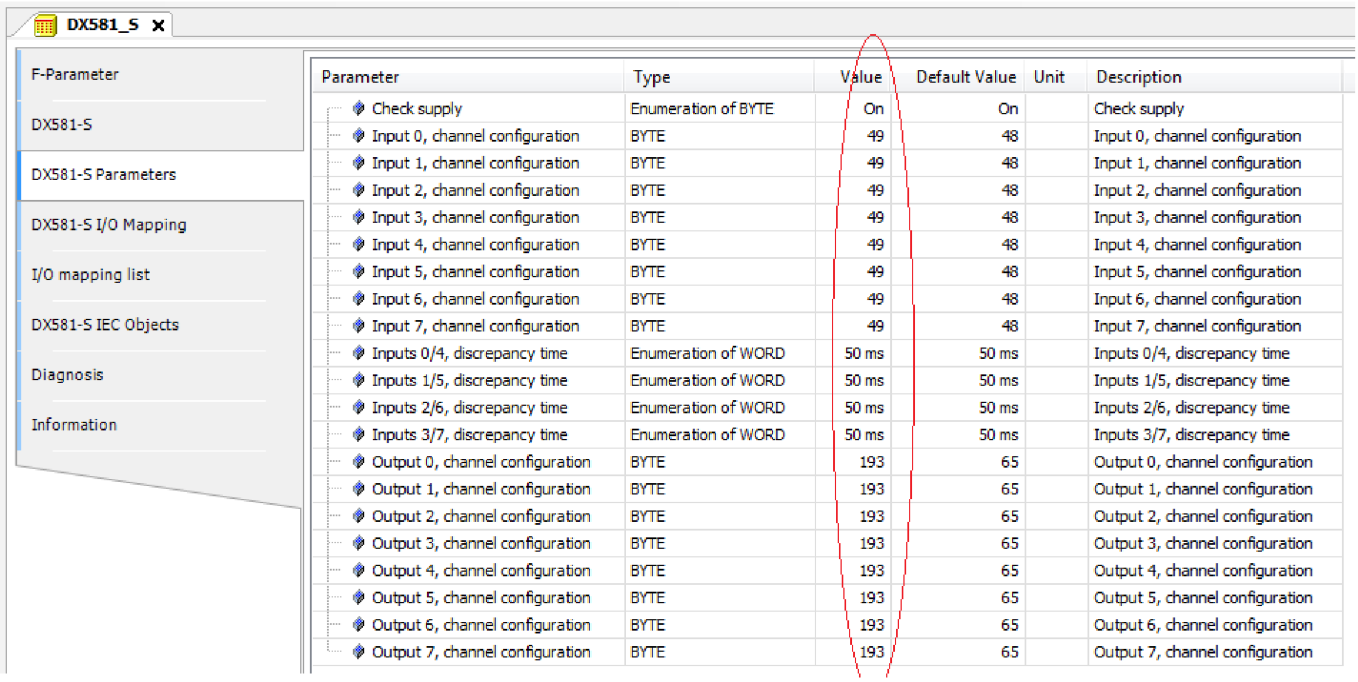
Personnel: ■ Safety application engineer of AC500-S safety PLC

1. In Automation Builder, go to *"Tools ➔ Options..."*. Activate *"Show generic device configuration views"* and instantiate a given type of safety I/O module (AI581-S, DI581-S or DX581-S) in the Automation Builder tree (DX581-S is used as an example):



2. Go to the iParameter setting tab (*"DX581-S"*, *"DI581-S"* or *"AI581-S"*) for the given module and set appropriate iParameter values (e.g., *"Test Pulse"*, *"Input Delay"*, etc.).
3. Verify against your safety application technical specification that all iParameters for all safety I/O channels are set correctly.
4. Go to *"F-Parameter"* tab and press *[Calculate]* button. Copy calculated F_iPar_CRC value from the *"Checksum iParameter"* field and paste it to *"F_iPar_CRC"* field of the F-Parameter editor.

5. Go to *“<safety I/O module name> Parameters”* tab, and verify using a cross-check according to *Chapter 6.5.2 “ Verification tables for iParameter settings in AC500-S safety I/Os” on page 342* that iParameter settings previously set at Step 2 are the same as ones listed in the *“Value”* column for given channels (use *Chapter 6.5.2 “ Verification tables for iParameter settings in AC500-S safety I/Os” on page 342* to decode integer values to real parameter values).



Parameter	Type	Value	Default Value	Unit	Description
Check supply	Enumeration of BYTE	On	On		Check supply
Input 0, channel configuration	BYTE	49	48		Input 0, channel configuration
Input 1, channel configuration	BYTE	49	48		Input 1, channel configuration
Input 2, channel configuration	BYTE	49	48		Input 2, channel configuration
Input 3, channel configuration	BYTE	49	48		Input 3, channel configuration
Input 4, channel configuration	BYTE	49	48		Input 4, channel configuration
Input 5, channel configuration	BYTE	49	48		Input 5, channel configuration
Input 6, channel configuration	BYTE	49	48		Input 6, channel configuration
Input 7, channel configuration	BYTE	49	48		Input 7, channel configuration
Inputs 0/4, discrepancy time	Enumeration of WORD	50 ms	50 ms		Inputs 0/4, discrepancy time
Inputs 1/5, discrepancy time	Enumeration of WORD	50 ms	50 ms		Inputs 1/5, discrepancy time
Inputs 2/6, discrepancy time	Enumeration of WORD	50 ms	50 ms		Inputs 2/6, discrepancy time
Inputs 3/7, discrepancy time	Enumeration of WORD	50 ms	50 ms		Inputs 3/7, discrepancy time
Output 0, channel configuration	BYTE	193	65		Output 0, channel configuration
Output 1, channel configuration	BYTE	193	65		Output 1, channel configuration
Output 2, channel configuration	BYTE	193	65		Output 2, channel configuration
Output 3, channel configuration	BYTE	193	65		Output 3, channel configuration
Output 4, channel configuration	BYTE	193	65		Output 4, channel configuration
Output 5, channel configuration	BYTE	193	65		Output 5, channel configuration
Output 6, channel configuration	BYTE	193	65		Output 6, channel configuration
Output 7, channel configuration	BYTE	193	65		Output 7, channel configuration

6. Go to *“F-Parameter”* tab and press *[Calculate]* button once more, even if the previous value is still available. Compare that the value shown in *“Checksum iParameter”* field and the one in F_iPar_CRC field of the F-Parameter editor are the same.

   ⇨ If F_iPar_CRC values are the same, then the verification procedure for given iParameter settings of the given AC500-S safety I/O module **was successfully passed.**

Important!

- If any errors (F_iPar_CRC or iParameters are not equal) were identified during Steps 1 ... 6, then one has to re-do the same procedure from the beginning. If after this second repetition there is still inconsistency, contact ABB technical support for help.
- Note, if iParameters values were verified as described in Steps 1 ... 6, you can re-use this iParameter combination with the given F_iPar_CRC for further modules of the same type without repeating the verification procedure described above.

6.5.2 Verification tables for iParameter settings in AC500-S safety I/Os

The instructions below provide a basis for cross-check of values set for iParameters in *“AI581-S”*, *“DI581-S”* and *“DX581-S”* tabs.

6.5.2.1 AI581-S safety I/O tables

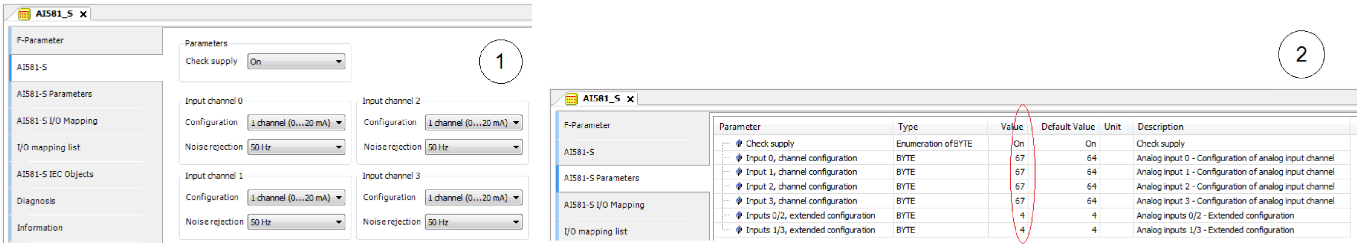


Fig. 125: The “AI581-S Parameters” tab is a readback view for iParameters set in “AI581-S” tab.

1 *“AI581-S”* tab
2 *“AI581-S Parameters”* tab

1. Compare that *“Check supply”* parameters have the same values "On" or "Off" in both *“AI581-S”* and *“AI581-S Parameters”* tabs.
2. Refer to *“AI581-S”* tab and calculate *“Input channel 0”* decimal equivalent **(Dec_InputChannel0)** as:

   Dec_InputChannel0 = Configuration_Value + Noise_Rejection_Value

   where:

   Configuration_Value:

   0 ➔ Not used

   3 ➔ 1 channel (0 ... 20 mA)

   4 ➔ 1 channel (4 ... 20 mA)

   5 ➔ 2 channel (4 ... 20 mA)

   Noise_Rejection_Value:

   0 ➔ None

   64 ➔ 50 Hz

   128 ➔ 60 Hz

   Compare calculated **Dec_InputChannel0** with **"Input 0, channel configuration value".** They have to be equal.

   If they are not equal, stop the procedure and re-do the configuration and comparison.

   If after the second iteration, there is still a difference between those values, stop verification procedure and contact ABB technical support.
3. Repeat step 2 for the rest of analog input channels (input 1, input 2 and input 3).

4. Refer to *"AI581-S"* tab and calculate **"Analog inputs 0/2 - Extended configuration"** decimal equivalent **(Dec_ExtConf0_2)** as:

   Dec_ExtConf0_2 = Tolerance_Range_Value + Min_Max_Value

   where

   Tolerance_Range_Value:

   4 ➔ 4 %

   5 ➔ 5 %

   6 ➔ 6 %

   7 ➔ 7 %

   8 ➔ 8 %

   9 ➔ 9 %

   10 ➔ 10 %

   11 ➔ 11 %

   12 ➔ 12 %

   Min_Max_Value:

   0 ➔ Min

   128 ➔ Max

   Compare calculated **Dec_ExtConf0_2** with **"Analog inputs 0/2 - Extended configuration".** They have to be equal.

   If they are not equal, stop the procedure and re-do the configuration and comparison.

   If after the second iteration, there is still a difference between those values, stop verification procedure and contact ABB technical support.

5. Repeat step 4 for "Analog inputs 1/3 - Extended configuration" value.

6.5.2.2 DI581-S safety I/O tables

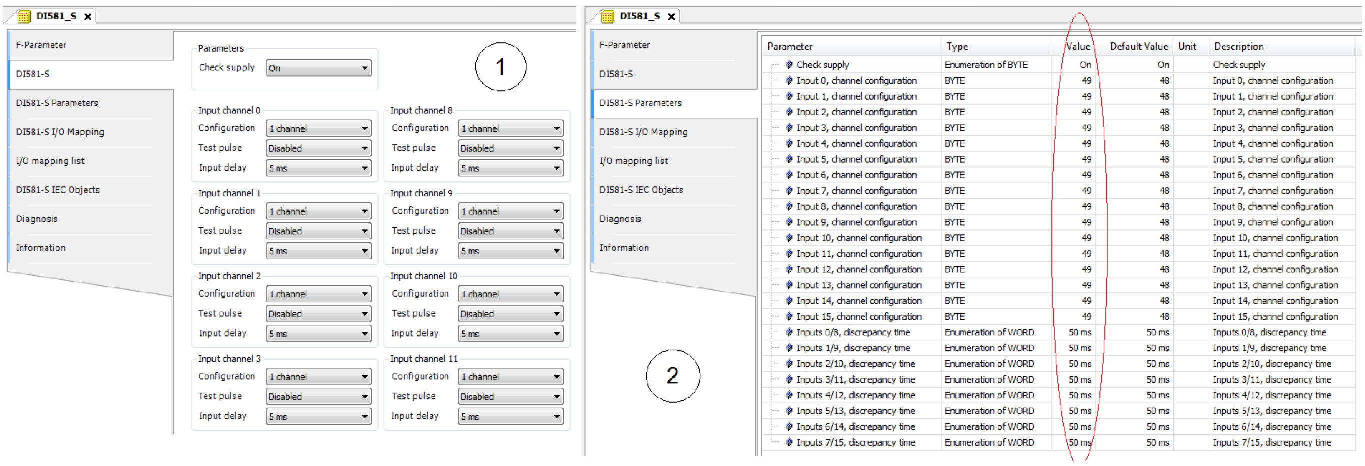


Fig. 126: The "DI581-S Parameters" tab is a readback view for iParameters set in "DI581-S" tab.

1 *"DI581-S"* tab
2 *"DI581-S Parameters"* tab

1. Compare that *"Check supply"* parameters have the same values "On" or "Off" in both *"DI581-S"* and *"DI581-S Parameters"* tabs.

2. Refer to *"DI581-S"* tab and calculate *"Input channel 0"* decimal equivalent **(Dec_InputChannel0)** as:

   Dec_InputChannel0 = Configuration_Value + Test_Pulse_Value + Input_Delay_Value

   where

   Configuration_Value:

   0 ➔ Not used

   1 ➔ 1 channel

   2 ➔ 2 channel equivalent

   3 ➔ 2 channel antivalent

   Test_Pulse_Value:

   0 ➔ Disabled

   8 ➔ Enabled

   Input_Delay_Value:

   16 ➔ 1 ms

   32 ➔ 2 ms

   48 ➔ 5 ms

   64 ➔ 10 ms

   80 ➔ 15 ms

   96 ➔ 30 ms

   112 ➔ 50 ms

   128 ➔ 100 ms

   144 ➔ 200 ms

   160 ➔ 500 ms

   Compare calculated **Dec_InputChannel0** with **"Input 0, channel configuration value".** They have to be equal.

   If they are not equal, stop the procedure and re-do the configuration and comparison.

   If after the second iteration, there is still a difference between those values, stop verification procedure and contact ABB technical support.

3. Repeat step 2 for the rest of digital input channels (input 1, input 2, ... input 15).

4. Compare that *“2 channel configuration 0/8”* parameter in *“DI581-S”* tab have the same value as *“Inputs 0/8, discrepancy time”* parameter in *“DI581-S Parameters”* tab.

   The values have to be the same.

   If they are not the same, stop the procedure and re-do the configuration and comparison.

   If after the second iteration, there is still a difference between those values, stop verification procedure and contact ABB technical support.

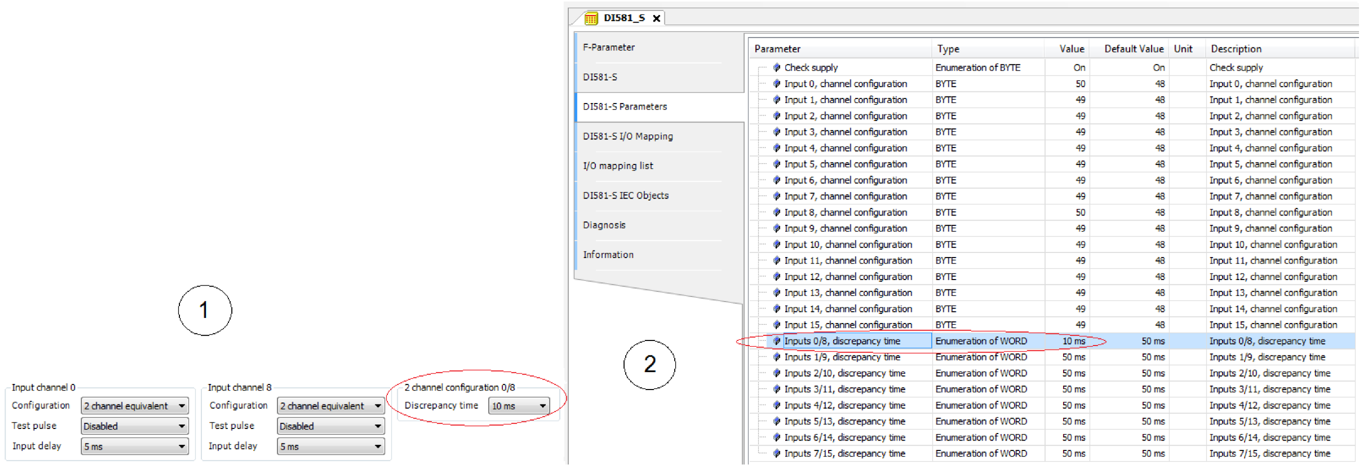


Fig. 127: Compare “DI581-S” tab and “DI581-S Parameters” tab

1 *“2 channel configuration 0/8”* parameter in *“DI581-S”* tab
2 *“Inputs 0/8, discrepancy time”* parameter in *“DI581-S Parameters”* tab

5. Repeat step 4 for the rest of channel combinations:

   - Inputs 1/9, discrepancy time
   - Inputs 2/10, discrepancy time
   - Inputs 3/11, discrepancy time
   - Inputs 4/12, discrepancy time
   - Inputs 5/13, discrepancy time
   - Inputs 6/14, discrepancy time
   - Inputs 7/15, discrepancy time

6.5.2.3 DX581-S safety I/O tables

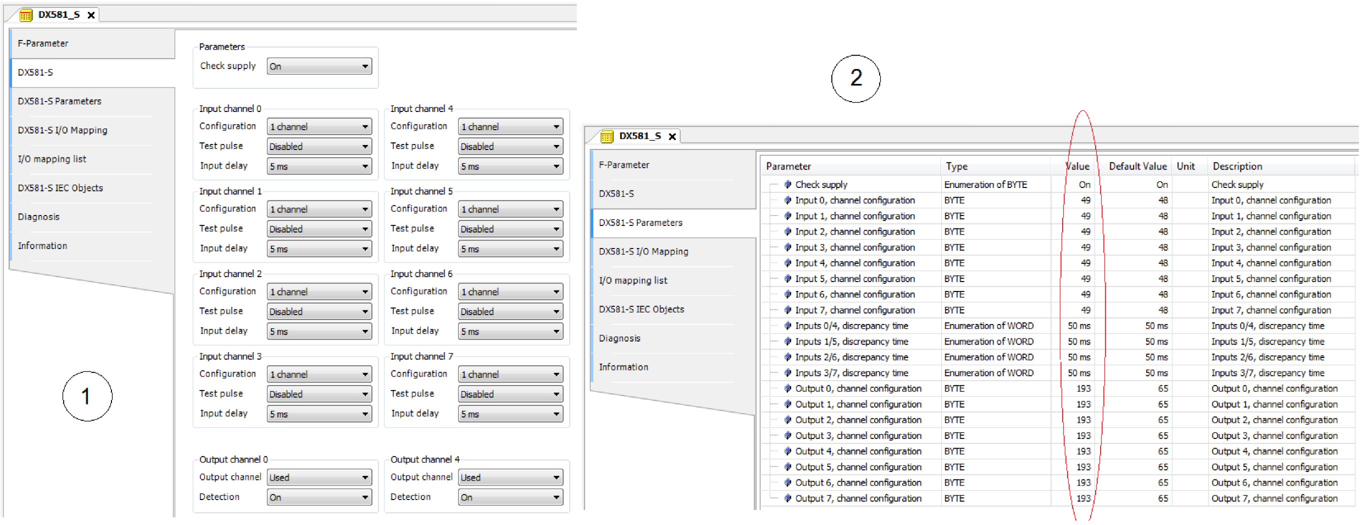


Fig. 128: The "DX581-S Parameters" tab is a readback view for iParameters set in "DX581-S" tab.

1 *"DX581-S"* tab
2 *"DX581-S Parameters"* tab

1. Compare that *"Check supply"* parameters have the same values "On" or "Off" in both *"DX581-S"* and *"DX581-S Parameters"* tabs.

2. Refer to *"DX581-S"* tab and calculate *"Input channel 0"* decimal equivalent **(Dec_InputChannel0)** as:

   Dec_InputChannel0 = Configuration_Value + Test_Pulse_Value + Input_Delay_Value

   where

   Configuration_Value:

   0 ➔ Not used

   1 ➔ 1 channel

   2 ➔ 2 channel equivalent

   3 ➔ 2 channel antivalent

   Test_Pulse_Value:

   0 ➔ Disabled

   8 ➔ Enabled

   Input_Delay_Value:

   16 ➔ 1 ms

   32 ➔ 2 ms

   48 ➔ 5 ms

   64 ➔ 10 ms

   80 ➔ 15 ms

   96 ➔ 30 ms

   112 ➔ 50 ms

   128 ➔ 100 ms

   144 ➔ 200 ms

   160 ➔ 500 ms

   Compare calculated **Dec_InputChannel0** with **"Input 0, channel configuration value".** They have to be equal.

   If they are not equal, stop the procedure and re-do the configuration and comparison.

   If after the second iteration, there is still a difference between those values, stop verification procedure and contact ABB technical support.

3. Repeat step 2 for the rest of digital input channels (input 1, input 2, ... input 7).

4. Compare that *"2 channel configuration 0/4"* parameter in *"DX581-S"* tab have the same value as *"Inputs 0/4, discrepancy time"* parameter in *"DX581-S Parameters"* tab.

   If they are not equal, stop the procedure and re-do the configuration and comparison.

   If after the second iteration, there is still a difference between those values, stop verification procedure and contact ABB technical support.

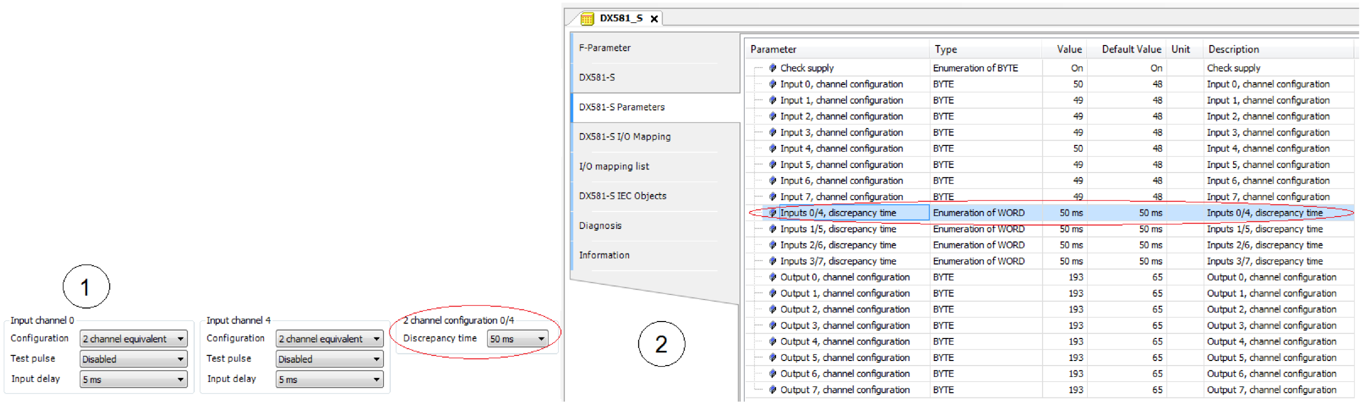


Fig. 129: Compare "DX581-S" tab and "DX581-S Parameters" tab

1 *"2 channel configuration 0/4"* parameter in *"DX581-S"* tab
2 *"Inputs 0/4, discrepancy time"* parameter in *"DX581-S Parameters"* tab

5. Repeat step 4 for the rest of input channel combinations:
   - Inputs 1/5, discrepancy time
   - Inputs 2/6, discrepancy time
   - Inputs 3/7, discrepancy time

6. Refer to *"DX581-S"* tab and calculate **"Output channel 0"** decimal equivalent (Dec_OutputChannel0) as:

   Dec_OutputChannel0 = Detection_Value + Output_Value + 1

   where

   Detection_Value:

   0 ➔ Off

   64 ➔ On

   Output_Value:

   0 ➔ Not used

   128 ➔ Used

   Compare calculated **Dec_OutputChannel0** with **"Output 0, channel configuration".** They have to be equal.

   If they are not equal, stop the procedure and re-do the configuration and comparison.

   If after the second iteration, there is still a difference between those values, stop verification procedure and contact ABB technical support.

7. Repeat step 6 for the rest of digital output channels (channel 1, channel 2, ... channel 7).

7 Safety application examples

7.1 Overview

In this chapter, application examples based on PLCopen Safety POUs are presented with the main goal to give an explanation on how PLCopen Safety POUs can be used in typical safety applications. Examples are used from ➪ *[7]* with a permission from PLCopen organization.

Initialization procedures for handling PROFIsafe start-up behavior and AC500-S specific POUs are not listed in these examples, but have to be included in the final safety application programs.

As an example of the usage of safety functions, the following production line is used. The PLCopen FBs described below can be used to easily realize the safety application program for this production line.

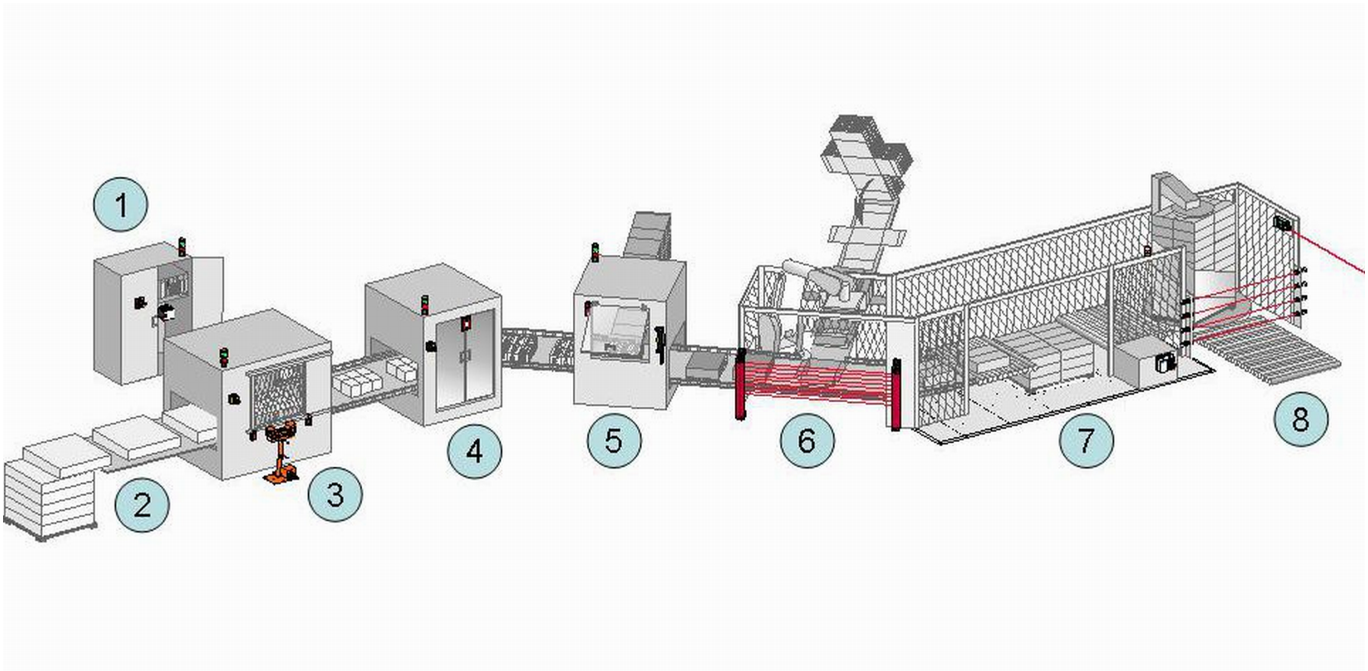


Fig. 130: Example of safety functionalities in a production line

1 Centralized control cabinet, including the safety related part of the control system where the safety related function blocks are running.
2 Infeed of material. In this part, no special safety related functions are used. However, safety functionalities like muting to separate between products and persons could be used.
3 Cutting of the material. For manual control, a two hand control safety function (unit is in front of the machine) is added combined with a 2-fold door monitoring system (attached to the door on the machine)
4 Automatic printing station, with door monitoring as safety function in case of service access (attached to the door on the machine)
5 First cartoning machine with door monitoring as safety function in case of service access (attached to the door on the machine). Sometimes, the manual operation is necessary. In this case, the operator can run the machine with a safely limited speed controlled by an enabling device which, when released, initiates a safe stop.
6 Second cartoning machine, guarded by an electro-sensitive protective equipment, ESPE. In this case, it is a light curtain.
7 Palletizing function, guarded by safety mats. This functionality could be coupled to the ESPE safety function.
8 Foil wrapping station of the palletized products with an exit of the production line. This area is safe-guarded by several combined light beams, coupled to the ESPE safety function.

In addition, every station is equipped with an emergency stop.

7.2 Example 1: diagnostics concept

This example shows the usage of the diagnostic concept, with a daisy chain from the FB parameters Activate and Ready (with perhaps a pre-evaluation of hardware errors). Other examples will not show the diagnostic connections *Chapter 7.3 “Example 2: muting” on page 355* *Chapter 7.4 “Example 3: two-hand control” on page 359*.

The safety functionality is to stop a drive in accordance with stop category 1 of IEC 60204-1 initiated by an emergency stop or by interrupting the light curtain. The equivalent monitoring of the 2 connectors of the emergency stop switch is done in the safety application.

In this example, both options of input evaluation are shown:

- via intelligent safety input
- via the equivalent function block

7.2.1 Functional description of safety functions

This example uses the following safety functions:

- Issuing the emergency stop (via SF_EmergencyStop) or interrupting the light beam in the light curtain (via SF_ESPE) stops the drive in accordance with stop category 1.
- The stop of the electrical drive within a predefined time is monitored (via SF_SafeStop1).
- The safe status of the drive is indicated by the S_Stopped variable, connected to the functional application.
- If the stop is performed by the emergency stop switch, a manual reset is required (via SF_EmergencyStop).
- If a monitoring time violation is detected (via SF_SafeStop1), manual error acknowledge is required to allow a reset.
- The 2 channel connectors of the emergency stop are monitored. An error is detected when both inputs do not have the same status once the discrepancy time has elapsed (via SF_Equivalent).
- The functional stop in this example is performed as a safe stop issued from the functional application. A restart interlock for this stop is not necessary.

7.2.2 Graphical overview of safety application interface

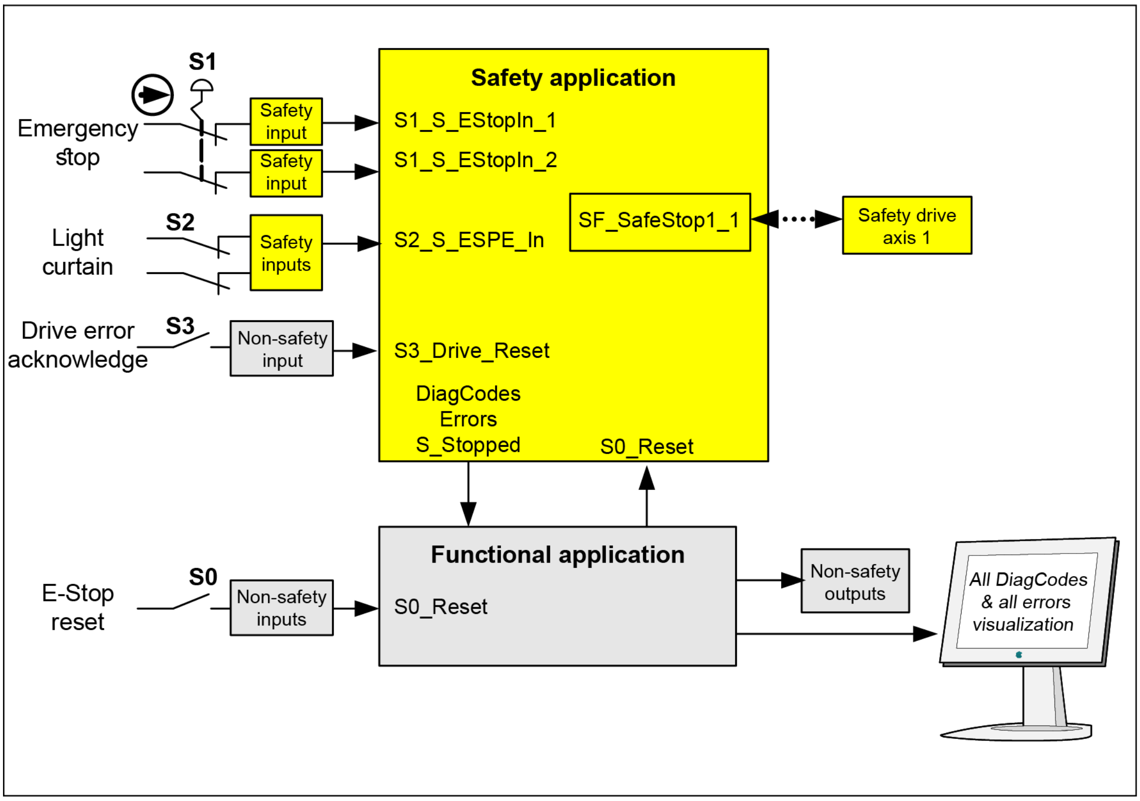


Fig. 131: Graphical overview of the example with emergency stop

⊛ The symbol represents a direct opening action (refer to IEC 60947-5-1).

7.2.3 Declaration of used variables

Table 93: Inputs

Name	Data type	Description
S1_S_EstopIn_1	BOOL	Emergency stop channel 1
S1_S_EstopIn_2	BOOL	Emergency stop channel 2
S2_ESPE_In	BOOL	Light curtain signal
S0_Reset	BOOL	Reset emergency stop and ESPE
S3_Drive_Reset	BOOL	Reset drive error

Table 94: Outputs

Name	Data type	Description
S_Stopped	BOOL	Indication of safe stop of drive
Errors	BOOL	Represents all errors of the used FB (connected to the functional application)
DiagCodes	WORD	Represents all diagnostic codes of the used FB (connected to the functional application)

Table 95: Hidden interface of FB instances towards drives (vendor specific)

Name	Description
SF_SafeStop1_1	Connection to Drive 1

Table 96: Local variable

Name	Data type	Description
S_EStopOut	BOOL	Emergency stop request
InputDevice1_active	BOOL	Status of the relevant input device as provided by the system
InputDevice2_active	BOOL	Status of the relevant input device as provided by the system

7.2.4 Program example

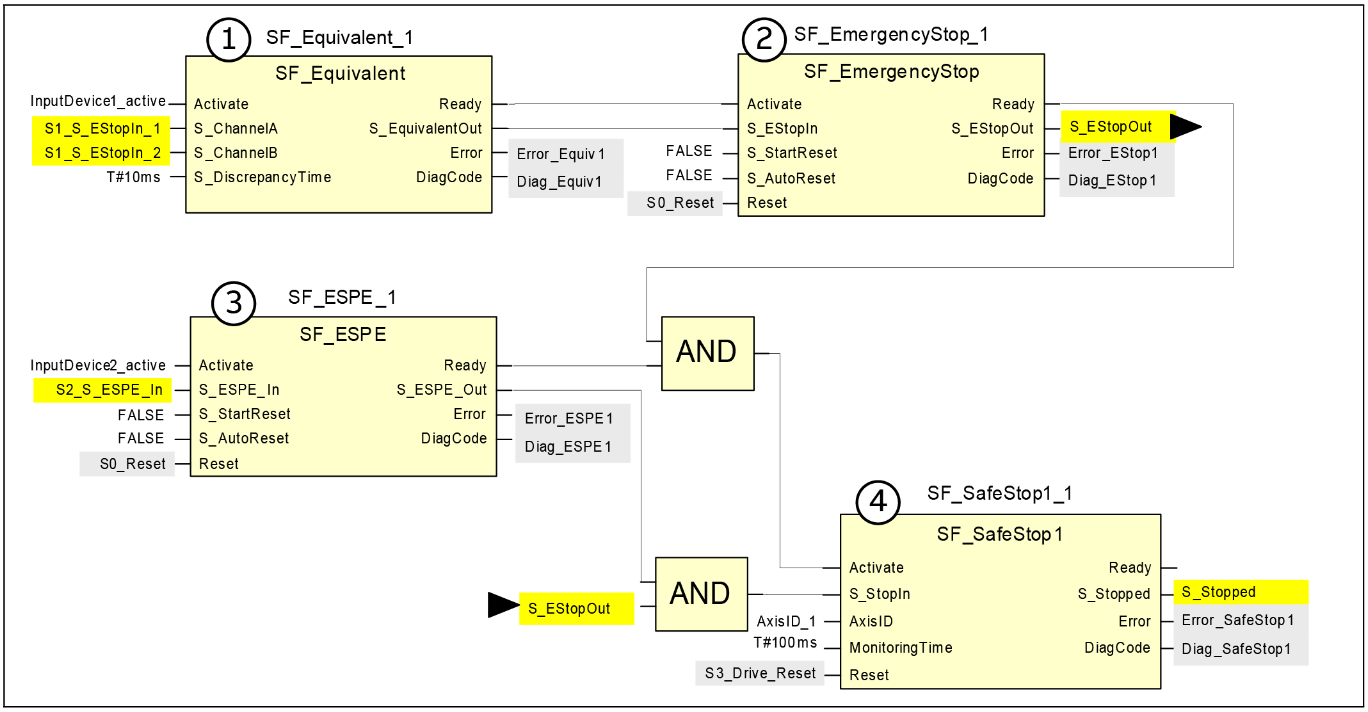


Fig. 132: Program example - emergency stop with safe stop and equivalence monitoring

1 Two channel line monitoring. Function block SF_Equivalent produces a single BOOL signal out of the two separated signals from the emergency stop channels. The discrepancy time is set constantly to 10 ms.
2 Emergency stop with restart inhibit. Function block SF_EmergencyStop handles the emergency stop condition. After the emergency stop request as well as after power up the safety output is only released after manual restart. This behavior is enabled by setting the S_StartReset and S_AutoReset inputs to FALSE.
3 ESPE: Function block SF_ESPE handles the light curtain interface. After intrusion in the protected field, as well as after power up, the safety output is only released after manual restart. This behavior is enabled by setting the S_StartReset and S_AutoReset inputs to FALSE.
4 Safe stop 1 request handling: Function block SF_SafeStop1 handles the safe stop 1 request for AxisID_1 and monitors that the axis follows the request within the predefined monitoring time of 100 ms. Any error condition within the axis has to be acknowledged by a manual drive reset signal.

7.2.5 Additional notes

This example uses different reset signals to acknowledge the emergency stop and to acknowledge the monitoring violation of the drive. If the safety requirement specification of the application allows the acknowledgment of both situations with the same signaling device, the identical signal from the functional application may be used to reset the FB SF_EmergencyStop_1 as well as to reset the FB SF_SafeStop1_1.

Information on the diagnostics concept

The representation of the diagnostics concept is for information only. For the safety functionality, the dedicated safety inputs and outputs shall be used.

Daisy chain from Activate and Ready

The connection of the Ready output to an Activate input of the following FB ensures that no irrelevant diagnostic information is generated if a device is disabled. The daisy chain from Activate and Ready avoid subsequent error messages of related function blocks.

Pre-evaluation of hardware errors

If the target system supports an error signal, e.g., InputDevice_active, which represents the status (active or not active) of the relevant safety device, this signal can be used to disable the safety function blocks. This ensures no irrelevant diagnostic information is generated if a device is disabled. If no such error signal is provided by the target system, a static TRUE signal must be assigned to the Activate input.

Evaluation of the diagnostic information

The Error signals and DiagCodes of each safety function block are transferred to the non-safety application. Diagnosis information might be processed and displayed by an attached visualization. There are different possibilities to realize the evaluation of the diagnostic information:

- Transfer these values into the visualization and realize the diagnostic evaluation in the visualization.
- Realize the diagnostic evaluation in the non-safety logic and transfer the results to the visualization.

Because of the various possibilities and the differences in the target system to realize diagnostic processing, there is no special example showed here. Further diagnostic processing could be:

- Display of the error status for each safety function block.
- Providing an error overview which is linked to function block specific error displays.
- Detection and display of the last error of the used safety function blocks in the safety application.

Information on the used function block parameters

Function block	Input	Constant value	Description
SF_Equivalent_1	S_DiscrepancyTime	10 ms	Maximum monitoring time for discrepancy status of both inputs.
SF_EmergencyStop_1	S_StartReset	FALSE	Manual reset when PES is started (warm or cold).
	S_AutoReset	FALSE	Manual reset when emergency stop button is released.
SF_SafeStop1_1	AxisID	AxisID_1	Drive address, supplier specific value
	MonitoringTime	100 ms	Time until the drive shall be stopped.
SF_ESPE	S_StartReset	FALSE	Manual reset when PES is started (warm or cold).
	S_AutoReset	FALSE	Manual reset after safety demand condition is cleared.

7.3 Example 2: muting

This example describes the safety functions for the safeguarding of a production cell. Objects are transferred through an entry gate, which is guarded by a light curtain. This light curtain can be muted only for material transport into the cell. The cell may be entered by the operator through a safety door. The process inside the cell is controlled by the functional application and enabled by the safety circuit. In case of a safety demand or an error, all hazardous movements are stopped in accordance with stop category 0.

7.3.1 Functional description of safety functions

All hazardous movements are stopped in case of:

- an opening of the door.
- an error (e.g., invalid muting sequence).
- an interruption of the unmuted light curtain (e.g., by a person).
- pushing an emergency stop button.

By pushing an emergency stop button, the operator can also stop all hazardous movements in stop category 0 (via SF_EmergencyStop and subsequent FBs).

An infringement of the unmuted light curtain stops all hazardous movements. In this application, a light curtain type 2 is used, which requires a test by the FB SF_TestableSafetySensor.

For the described muting function, four muting sensors are applied sequentially (via SF_MutingSeq). Additionally, the muting phase is indicated by a lamp, which is monitored in this case (also via SF_MutingSeq).

An additional door for maintenance purposes is monitored by a door switch (via SF_GuardMonitoring).

By resetting buttons, the operator must acknowledge the detected demand of the safety functions and errors.

The initial state and the operational state of the connected actuator are checked by an external device monitoring. In case an error is detected, the control cannot become operational (via SF_EDM).

The process and related movements inside the production cell are controlled by the functional application. Within the safety application, this control is enabled by the above-described safety circuit (via SF_OutControl) and drives the actuator via a safety output.

7.3.2 Graphical overview of the safety application interface

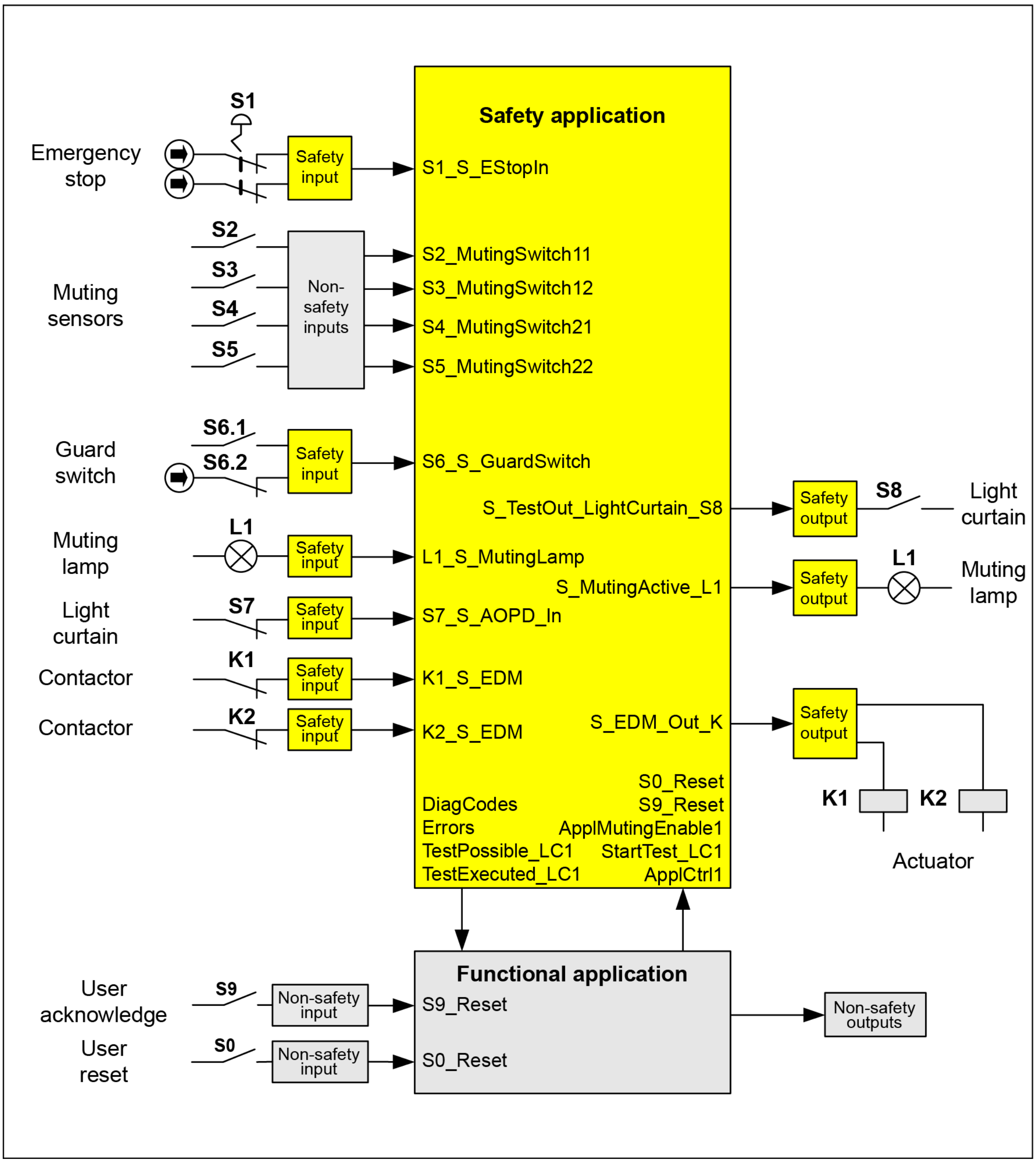


Fig. 133: Graphical overview of the exemplary access protection at a material gate

7.3.3 Declaration of used variables

Table 97: Inputs

Name	Data type	Description
S1_S_EStopIn	BOOL	Emergency stop button S1
S2_MutingSwitch11	BOOL	Muting sensor S2
S3_MutingSwitch12	BOOL	Muting sensor S3
S4_MutingSwitch21	BOOL	Muting sensor S4
S5_MutingSwitch22	BOOL	Muting sensor S5
S6_S_GuardSwitch	BOOL	Door switch S6 with two contacts

Name	Data type	Description
L1_S_MutingLamp	BOOL	Muting lamp monitor signal L1
S7_S_AOPD_In	BOOL	OSSD from light curtain S7
K1_S_EDM	BOOL	Feedback from external device K1 (actuator)
K2_S_EDM	BOOL	Feedback from external device K2 (actuator)
S9_Reset	BOOL	Reset safety demand by user S9
S0_Reset	BOOL	Reset error by user S0 (derived from functional application)
ApplCtrl1	BOOL	Signal controlling the actuator, enabled by safety loop (derived from functional application)
StartTest_LC1	BOOL	Signal starting test of light curtain S7 (derived from functional application)
ApplMutingEnable1	BOOL	Signal enabling start of the muting sequence (derived from functional application)

Table 98: Outputs

Name	Data type	Description
S_EDM_Out_K	BOOL	Drives actuator via K1 and K2
S_MutingActive_L1	BOOL	Drives muting lamp L1
S_TestOut_LightCurtain_S8	BOOL	Test output for light curtain S8
Errors	BOOL	Represents all errors of the used FB (connected to functional application)
DiagCodes	WORD	Represents all diagnostic codes of the used FB (connected to functional application)
TestPossible_LC1	BOOL	Indicates to the functional application that an automatic sensor test of the light curtain is possible.
TestExecuted_LC1	BOOL	Indicates to the functional application the successful execution of an automatic sensor test of the light curtain.

Table 99: Local variables

Name	Data type	Description
S_SafeControl	BOOL	Indicates the status of the safety guards (TRUE = safety enabled)

7.3.4 Program example

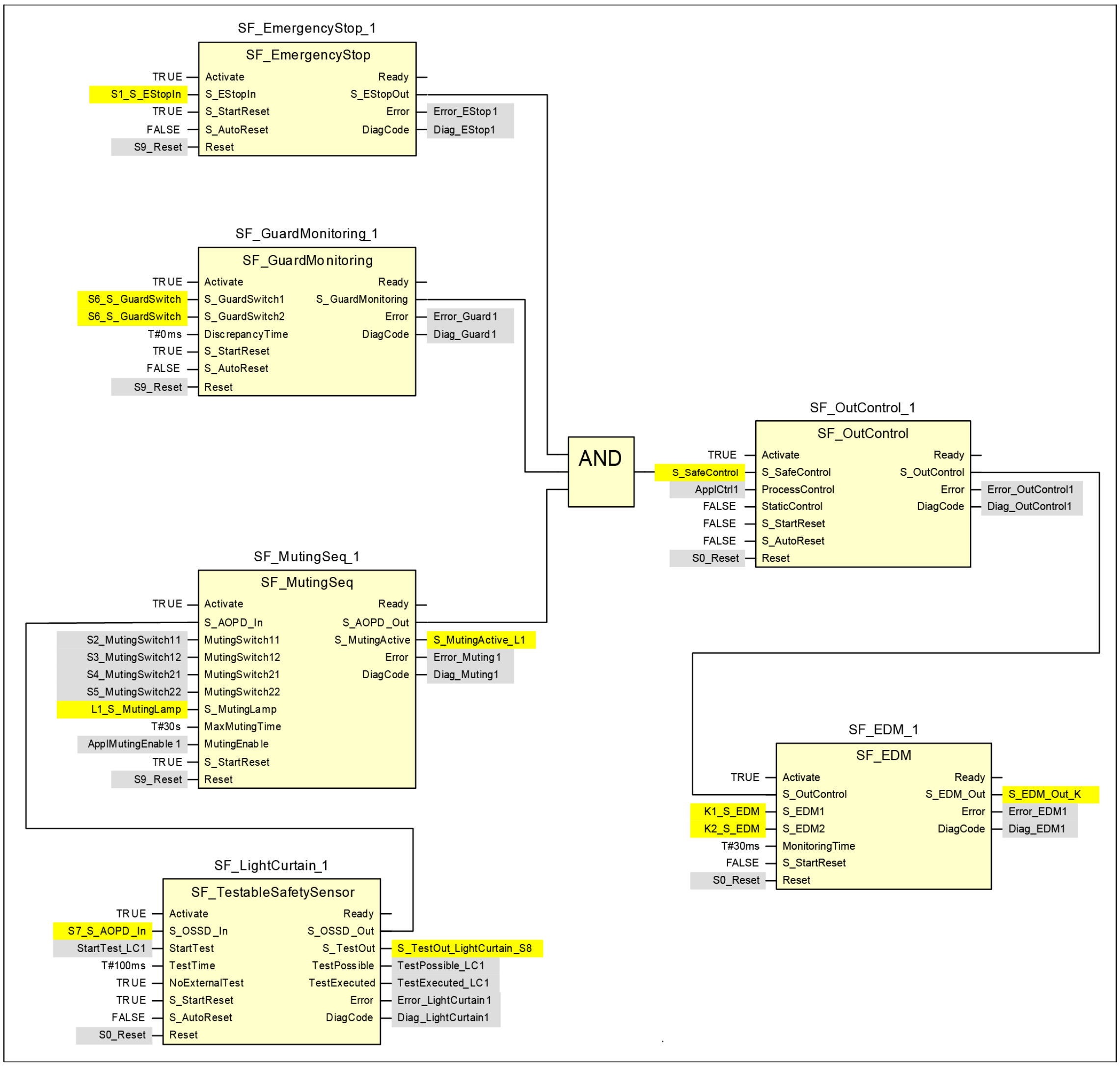


Fig. 134: Access protection at a material gate

7.3.5 Additional notes

In this example, two contacts of the guard switch are connected to a safety input device, which realizes the error detection. The resulting BOOL signal is mapped to the two input channels of the SF_GuardMonitoring_1.

The diagnostic information retrieval has not been covered in this example. For this, refer to *Chapter 7.2.5 “Additional notes” on page 354*. The input parameter Activate for the dynamic FB activation has been set to TRUE. However, in an application, this can be replaced by a variable.

Information about the used function block parameters

Function block	Input	Constant value	Description
SF_EmergencyStop_1	S_StartReset	TRUE	Automatic reset allowed when PES is started
	S_AutoReset	FALSE	No automatic reset, user reset/acknowledge necessary

Function block	Input	Constant value	Description
SF_GuardMoni-toring_1	S_StartReset	TRUE	Automatic reset allowed when PES is started
	S_AutoReset	FALSE	No automatic reset, user reset/acknowledge necessary
	DiscrepancyTime	T#0ms	The discrepancy time between both safety inputs S_GuardSwitchX is not monitored, because they are identical and since the input unit provides one signal of type BOOL from the contactors.
SF_MutingSeq_1	S_StartReset	TRUE	Automatic reset allowed when PES is started
	MaxMutingTime	T#30s	The maximum muting time is monitored to be within 30 s
SF_LightCurtain_1	S_StartReset	TRUE	Automatic reset allowed when PES is started
	S_AutoReset	FALSE	No automatic reset, user reset/acknowledge necessary
	TestTime	T#100ms	The maximum test time is monitored to be within 100 ms
	NoExternalTest	TRUE	The external manual sensor test is not sup-ported.
SF_OutControl_1	S_StartReset	FALSE	No automatic reset allowed when PES is started
	S_AutoReset	FALSE	No automatic reset, user reset/acknowledge necessary
	StaticControl	FALSE	A dynamic change of the signal ApplCtrl1 (rising edge) is required after block activation or a triggered safety function (S_SafeControl = FALSE).
SF_EDM_Contactor_1	S_StartReset	FALSE	No automatic reset allowed when PES is started
	MonitoringTime	T#30ms	The maximum response time of both feed-back signals S_EDM1 and S_EDM2 are monitored to be within 30 ms.

7.4 Example 3: two-hand control

This example describes a machine where a two-hand control initiates the dangerous movement as long as both push buttons on the two-hand control are pressed and the process provides an enabling signal.

The dangerous movement is initiated by the closing of two subsequent contactors, which are monitored via a feedback loop.

7.4.1 Functional description of safety functions

This example uses the following safety functions:

- By pushing an emergency stop button all hazardous movements must be stopped (via SF_EmergencyStop). Emergency stop has the highest priority. After releasing the E-Stop push button, a reset via S0_Reset is required.
- By pressing both push buttons of the two-hand control, the safety output is activated. The release of any of the two-hand push buttons disables the safety output and stops the dan-gerous motion via the contactors K1 and K2 (via SF_TwoHandControlTypeII).

- The initial state and the operational state of the connected contactors K1 and K2 are monitored and if an error is detected, the safety output cannot become operational (via SF_EDM).
- After power-on of the safety or functional application, or after an emergency stop condition, the two-hand control must be released and re-operated in order to activate the safety output again (via SF_OutControl). In order to guarantee this for the functional application restart, the process signal from the functional application is connected to the Activate input of the two hand control function block THC_S2_S3 (If the application process is restarted while the two hand control is activated, the FB goes to the state C003 signaling an error that both buttons are pressed at the activation, prohibiting a restart.).

In this example, only one operation mode exists.

7.4.2 Graphical overview of the safety application interface

The safety inputs for the two-hand control (S2_S_Switch1 and S3_S_Switch2) are connected to the two-hand control type II.

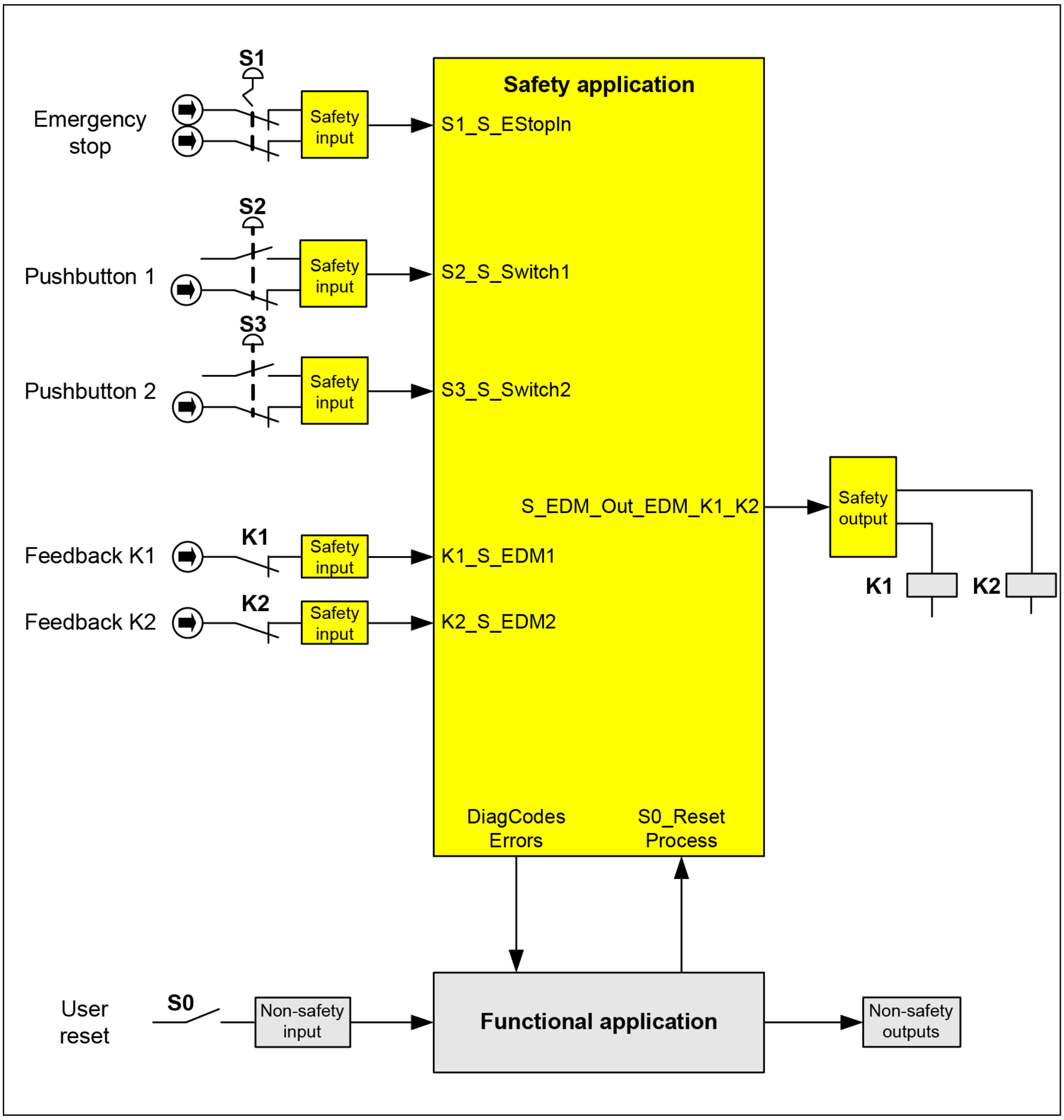


Fig. 135: Graphical overview of the exemplary two-hand control with EDM

7.4.3 Declaration of used variables

Table 100: Inputs

Name	Data type	Description
S1_S_EStopIn	BOOL	Emergency stop button S1
S2_S_Switch1	BOOL	Switch S2 related to push button 1 of two hand control
S3_S_Switch2	BOOL	Switch S3 related to push button 2 of two hand control
K1_S_EDM1	BOOL	Feedback from external device K1 (actuator)
K2_S_EDM2	BOOL	Feedback from external device K2 (actuator)
S0_Reset	BOOL	Reset by user via switch S0 (derived from the functional application)
Process	BOOL	Enabling motion by the process (derived from functional application)

Table 101: Outputs

Name	Data type	Description
S_EDM_Out_EDM_K1_K2	BOOL	Drives actuator via K1 and K2
Errors	BOOL	Represents all errors of the used FB (connected to the functional application)
DiagCodes	WORD	Represents all diagnostic codes of the used FB (connected to the functional application)

7.4.4 Program example

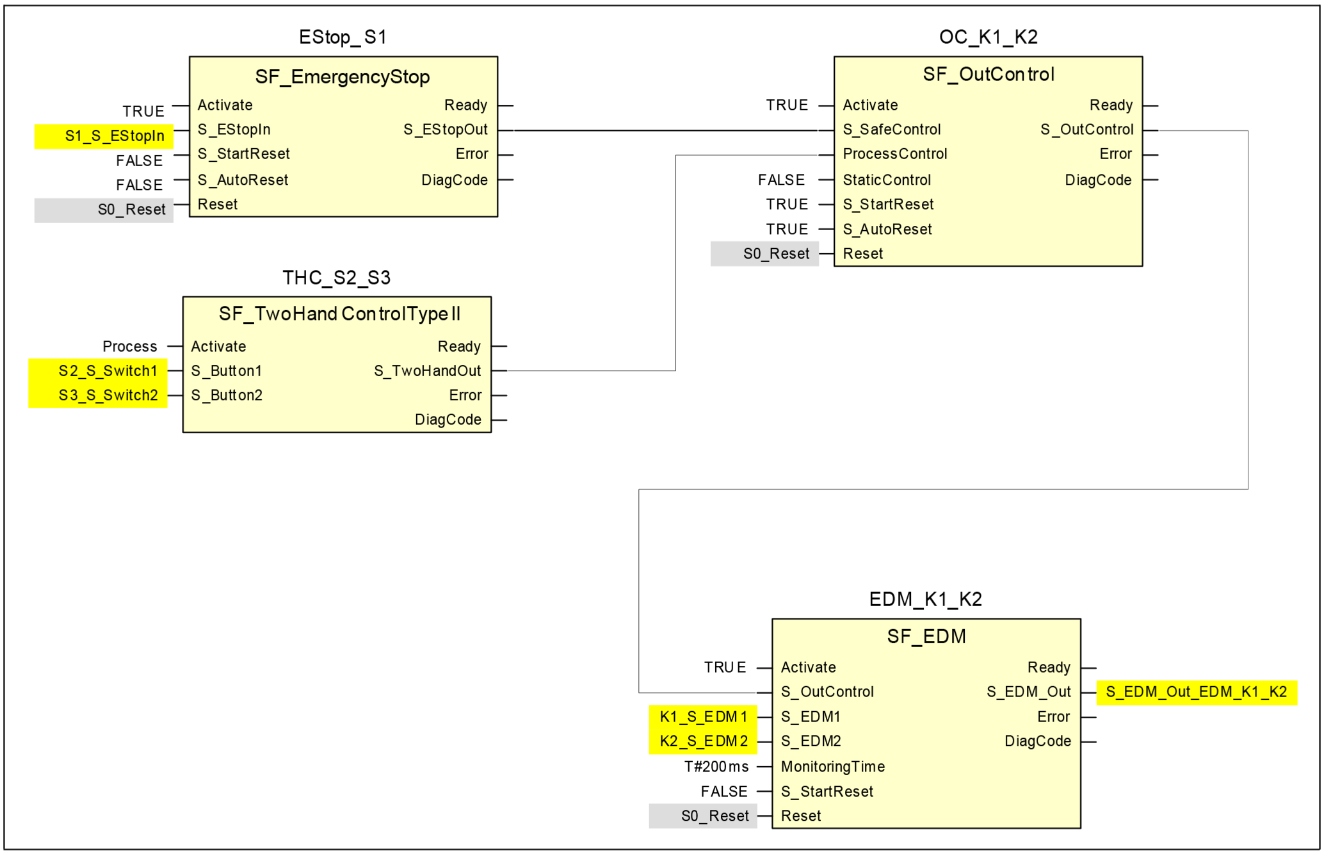


Fig. 136: Application program of two-hand control with EDM

7.4.5 Additional notes

This example can also be used with the SF_TwoHandControlTypeIII.

The diagnostic information retrieval has not been covered in this example. For this, refer to ➯ *Chapter 7.2.5 “Additional notes” on page 354*. The input Activate has been set to TRUE. However, in an application this can be replaced by a variable.

Information about the used function block parameters

Function block	Input	Constant value	Description
EStop_S1	S_StartReset	FALSE	No automatic reset when PES is started
	S_AutoReset	FALSE	No automatic reset, user reset/acknowledge necessary
OC_K1_K2	S_StartReset	TRUE	Automatic reset allowed when PES is started
	S_AutoReset	TRUE	Automatic reset, no user reset/acknowledge necessary
	StaticControl	FALSE	A dynamic change of the signal Appl_Control (rising edge) is required after block activation or a triggered safety function (S_SafeControl = FALSE)
EDM_K1_K2	S_StartReset	FALSE	No automatic reset when PES is started
	MonitoringTime	T#200ms	The maximum response time of both feed-back signals S_EDM1 and S_EDM2 is moni-tored to be within 200 ms.

8 Index

V

W

Appendix

A System data for AC500-S-XC

A.1 Environmental conditions

Process and supply voltages

Data	Value	Unit
Process and supply voltage (-25 %, +30 % inclusive ripple)	24	V DC
Absolute limits inclusive ripple	18 ... 31.2	V DC
Ripple	< 10	%
Protection against reverse polarity	yes	
Allowed interruptions of DC power supply	< 10	ms
Time between 2 interruptions, PS2	> 1	s

DANGER!
Exceeding the permitted process or supply voltage range (< -35 V DC or > +35 V DC) could lead to unrecoverable damage of the system.

DANGER!
For the supply of the modules, power supply units according to PELV or SELV specifications must be used.

NOTICE!
The creepage distances and clearances meet the requirements of the over-voltage category II, pollution degree 2.

Temperature

Data	Value	Unit
Operating temperature*	-40 ... +70	°C
Operating temperature (vertical mounting of module output load limited to 50 % per group)	-40 ... +40	°C
Storage temperature	-40 ... +85	°C
Transport temperature	-40 ... +85	°C

* +60 ... +70 °C with the following deratings:

- Terminal bases: Maximum 2 communication modules allowed.
- Digital inputs: Maximum number of simultaneously switched on input channels limited to 50 % per group (e.g. 8 channels => 4 channels).
- Digital outputs: Output current maximum value (all channels together) limited to 50 % per group (e.g. 4 A => 2 A).
- Analog inputs: No limitations.

DANGER!
The average temperature (MTBF calculation base) for both the extended temperature range (-40 ... +70 °C) as well as for normal temperature range (0 ... +60 °C) is defined to +40 °C.

Ensure that average operating temperature for used AC500-S-XC modules does not exceed +40 °C.

Humidity

Data	Value	Unit
Relative humidity with condensation (operating/storage)	100	%

Air pressure

Data	Value	Unit
Operating air pressure	1080 ... 620	hPa
Operating altitude	-1000 ... 4000	m
Reduction of operating temperature at an air pressure of < 795 hPa (or > 2000 m above sea level)	10 (e.g. +70 °C to +60 °C)	K

Immunity to corrosive gases

Data	Value
Operating: according to ISA S71.04.1985 harsh group A, G3/GX IEC 60721-3-3 3C2 / 3C3	yes

Immunity to salt mist

Data	Value
Operating: horizontal mounting only, according to IEC 60068-2-52 severity level 1	yes

Electromagnetic compatibility

Data	Value
Radiated emission (radio disturbance) according to CISPR 16-2-3	yes
Conducted emission (radio disturbance) according to CISPR 16-2-1, CISPR 16-1-2	yes
Electrostatic discharge (ESD) according to IEC 61000-4-2, zone B, criterion B	yes
Fast transient interference voltages (burst) according to IEC 61000-4-4, zone B, criterion B	yes
High energy transient interference voltages (surge) according to IEC 61000-4-5, zone B, criterion B	yes
Influence of radiated disturbances according to IEC 61000-4-3, zone B, criterion A	yes
Influence of line-conducted interferences according to IEC 61000-4-6, zone B, criterion A	yes
Influence of power frequency magnetic fields according to IEC 61000-4-8, zone B, criterion A	yes

> **NOTICE!**
> In order to prevent malfunctions, it is recommended that the operating personnel discharge themselves prior to touching communication connectors or perform other suitable measures to reduce effects of electrostatic discharges.

> **NOTICE!**
> Unused sockets for communication modules on terminal bases must be covered with TA524 dummy communication module. I/O bus connectors must not be touched during operation.

Radiation

Data	Value
Radio disturbance according to IEC 55011, group 1, class A	yes

A.2 Mechanical data

Data	Value
Wiring method	spring terminals
Degree of protection	IP 20
Vibration resistance according to IEC 61131-2, IEC 60068-2-6, IEC 60068-2-64	yes
Shock resistance according to IEC 60068-2-27	yes
Horizontal assembly position	yes
Vertical assembly position (no application in salt mist environment)	yes

Assembly on DIN rail according to IEC 60715

Data	Value	Unit
DIN rail type	35	mm
DIN rail type depth	7.5 or 15	mm

Assembly with screws

Data	Value	Unit
Screw diameter	4	mm
Fastening torque	1.2	Nm

A.3 Environmental tests

Storage	IEC 60068-2-1 test Ab: cold withstand test -40 °C / 16 h IEC 60068-2-2 test Bb: dry heat withstand test +85 °C / 16 h
Humidity	IEC 60068-2-30 test Dd: Cyclic (12 h / 12 h) damp-heat test +55 °C, 93 % relative humidity / +25 °C, 95 % relative humidity, 6 cycles IEC 60068-2-78, stationary humidity test: +40 °C, 93 % relative humidity, 240 h
Insulation test	IEC 61131-2
Vibration resistance	IEC 61131-2 / IEC 60068-2-6: 5 Hz ... 500 Hz, 2 g (with SD memory card inserted in non-safety CPU) IEC 60068-2-64: 5 Hz ... 500 Hz, 4 g rms
Shock resistance	IEC 60068-2-27: all 3 axes 15 g, 11 ms, half-sinusoidal

EMC immunity

Electrostatic discharge (ESD)

Data	Value	Unit
Electrostatic voltage in case of air discharge	8	kV
Electrostatic voltage in case of contact discharge	6	kV

Fast transient interference voltages (burst)

Data	Value	Unit
Supply voltage units (DC)	4	kV
Digital inputs/outputs (24 V DC)	2	kV
Analog inputs/outputs	2	kV
Communication lines, shielded	2	kV
I/O supply (DC-out)	2	kV

High energy transient interference voltages (surge) - common mode (CM)

Data	Value	Unit
Supply voltage units (DC)	1	kV
Digital inputs/outputs (24 V DC)	1	kV
Analog inputs/outputs	1	kV
Communication lines, shielded	1	kV
I/O supply (DC-out)	0.5	kV

High energy transient interference voltages (surge) - differential mode (DM)

Data	Value	Unit
Supply voltage units (DC)	0.5	kV
Digital inputs/outputs (24 V DC)	0.5	kV
Analog inputs/outputs	0.5	kV
I/O supply (DC-out)	0.5	kV

Data	Value	Unit
Influence of radiated disturbances: test field strength	10	V/m
Influence of line-conducted interferences: test voltage	10	V

Data	Value	Unit
Power frequency magnetic fields at 30 A/m	50 and 60	Hz

NOTICE!

Extreme environmental conditions and relevant requirements for used non-safety CPUs and I/O modules from AC500-XC family shall be taken into account ↳ *[4]*.

B Usage of safety CPU with AC500 V2 non-safety CPU PM5xx

B.1 Compatibility with AC500 V2 non-safety CPU

All compatibility information is valid for normal and XC devices.

Table 102: Compatibility for safety CPU with AC500 V2 non-safety CPU

Safety CPU	**SM560-S**	**SM560-S-FD-1, SM560-S-FD-4**
Firmware version of safety CPU	Any	V2.0.0 or higher
Non-safety CPU	Any V2 CPU, except AC500-eCo CPUs	Any V2 CPU, except AC500-eCo CPUs
Firmware version of non-safety CPU	V2.2.1 or higher	V2.7 or higher
Version of engineering suite Automation Builder	1.0 or higher	2.1 or higher
Version of engineering suite Control Builder Plus	V2.2.1 or higher	Not compatible

Table 103: Compatibility for AC500-S with non-safety components except CPUs

Component	**SM560-S**	**SM560-S-FD-1, SM560-S-FD-4**
Firmware version of communication module CM579-PNIO	V2.6.5.1 or higher	V2.6.5.1 or higher
Firmware version of communication module CM589-PNIO(-4)	Not applicable	V1.6.2.20 or higher
Fimware version of communication interface module CI501-PNIO, CI502-PNIO, CI504-PNIO, CI506-PNIO	V3.2.0 or higher	V3.2.0 or higher

B.2 Error messages with AC500 V2 non-safety CPU

NOTICE!

The error messages of the safety CPU are aggregated in the diagnosis stack on non-safety CPU.

You can use `diagreset`, `diagack all`, `diagack x`, `diagshow all` and `diagshow x` commands in non-safety CODESYS PLC browser to list and process various error messages in AC500 system, including those in the safety CPU. More details on these commands can be found in ⇘ *[4]*.

Using CM589-PNIO or CM589-PNIO-4 IO device communication modules, one can also generate PROFINET diagnostic messages for F-Devices of SM560-S-FD-1 and SM560-S-FD-4 ⇘ *Table 105 “Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs ” on page 378* ⇘ *Table 106 “Mapping of AC500/AC500-S errors to PROFINET channel errors” on page 379*.

B.2.1 Error messages for safety CPUs

The errors are shown as they are displayed in Automation Builder.

Table 104: Common error messages for SM560-S / SM560-S-FD-1 / SM560-S-FD-4 safety CPUs

Error severity	Compo-nent or interface	Device	Module	Channel	Error	Error text	Remedy
E2	1 ... 4	255	30	1	0	Operation fin-ished.	Change safety PLC switch address setting or remove SD-card from non-safety PLC. Restart safety PLC. If this error persists, replace safety PLC.
E2	1 ... 4	255	30	1	1	Wrong user data	Delete user data from safety PLC. Restart safety PLC and write user data again.
E2	1 ... 4	255	30	1	2	Internal PROFIsafe initi-alization error	Restart safety PLC. If this error persists, replace safety PLC. Contact ABB technical support.
E2	1 ... 4	255	30	1	12	Flash read error	Restart safety PLC. If this error persists, replace safety PLC. Contact ABB technical support.
E2	1 ... 4	255	30	1	18	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	1	28	Boot project download error	Reload boot project. If this error persists, replace safety PLC.
E2	1 ... 4	255	30	1	40	Wrong firmware version	Update safety PLC firmware. Restart safety PLC. If this error persists, replace safety PLC.
E2	1 ... 4	255	30	1	43	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	1	48	Overvoltage or undervoltage detected	Restart safety PLC. Check safety PLC setting for power supply error. If this error per-sists, replace safety PLC.
E2	1 ... 4	255	30	1	52	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	0	User program triggered safe stop	Check user program
E2	1 ... 4	255	30	2	1	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	2	Internal PROFIsafe error	Restart safety PLC. If this error persists, replace safety PLC. Contact ABB technical support.
E2	1 ... 4	255	30	2	3	Internal error	Contact ABB technical sup-port. Replace safety PLC.

Error severity	Compo-nent or interface	Device	Module	Channel	Error	Error text	Remedy
E2	1 ... 4	255	30	2	10	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	13	Flash write error	Restart safety PLC. If this error persists, replace safety PLC. Contact ABB technical support.
E2	1 ... 4	255	30	2	17	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	18	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	19	Checksum error has occurred in safety PLC.	Restart safety PLC. If this error persists, replace safety PLC.
E2	1 ... 4	255	30	2	25	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	37	Cycle time error in safety PLC	Check safety PLC watchdog time.
E2	1 ... 4	255	30	2	38	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	42	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	43	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	52	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	2	54	Internal error	Contact ABB technical sup-port. Replace safety PLC.
E2	1 ... 4	255	30	3	30	PROFIsafe con-figuration error	Check F-Parameter configu-ration of I/O module and reload boot project.
E2	9	1 ... 4	1	0	17	Access test failed	Check safety PLC switch address setting. Restart safety PLC. If this error per-sists, replace safety PLC.
E2	9	1 ... 4	1	0	43	Internal error	Check safety PLC switch address setting. Restart safety PLC. If this error per-sists, replace safety PLC
E2	9	1 ... 4	31	0	43	Internal error	Replace module
E3	1 ... 4	255	30	1	26	Error in configu-ration data, safety PLC cannot read configuration data	Create new configuration data
E3	1 ... 4	255	30	1	27	Error in configu-ration data, safety PLC cannot read configuration data	Create boot project

Error severity	Compo- nent or interface	Device	Module	Channel	Error	Error text	Remedy
E4	1 ... 4	255	30	1	0	Reserved switch address setting.	Warning
E4	1 ... 4	255	30	1	4	Boot project not loaded, max- imum power dip reached	Restart safety PLC
E4	1 ... 4	255	30	1	8	Power dip data missed or cor- rupted. Default power dip data was flashed by safety PLC.	Warning
E4	1 ... 4	255	30	1	19	CRC error boot project	Create new boot project and restart safety PLC
E4	1 ... 4	255	30	2	13	Flash write error (production data)	Warning
E4	1 ... 4	255	30	2	26	No or wrong configuration data from PM5x, run state not possible	Create correct boot project at PM5x
E4	1 ... 4	255	30	2	39	More than one instance of SF_WDOG_TIME_SET or SF_MAX_POWER_DIP_SET	Warning
E4	1 ... 4	255	30	4	13	Flash write error (boot project)	Warning
E4	1 ... 4	255	30	5	13	Flash write error (boot code)	Warning
E4	1 ... 4	255	30	6	13	Flash write error (firmware)	Warning
E4	1 ... 4	255	30	7	13	Flash write error (password)	Warning
E4	1 ... 4	255	30	8	13	Flash write error (user data)	Warning
E4	1 ... 4	255	30	9	13	Flash write error (user data)	Warning
E4	1 ... 4	255	30	10	13	Flash write error (internal)	Warning
E4	1 ... 4	255	30	11	13	Flash write error (internal)	Warning
E4	1 ... 4	255	30	12	13	Flash write error (internal)	Warning

Table 105: Specific error messages for SM560-S-FD-1 / SM560-S-FD-4 safety CPUs

Error severity	Compo-nent or interface	Device	Module	Channel	Error	Error text	Remedy
E2	1 ... 4	255	28	0 ... 31	43	Internal PROFIsafe F-Device error	Restart safety PLC. If this error persists, replace safety PLC. Contact ABB technical support.
E3	1 ... 4	255	28	0 ... 31	1	Safety destination address not valid (F_Dest_Add)	Check safety PLC configuration or switch address setting. Restart safety PLC. If this error persists, replace safety PLC.
E3	1 ... 4	255	28	0 ... 31	2	Safety source address not valid (F_Source_Add)	Check safety PLC configuration.
E3	1 ... 4	255	28	0 ... 31	10	Parameter "F_SIL" exceeds SIL from specific device application	Check safety PLC configuration.
E3	1 ... 4	255	28	0 ... 31	11	Safety watchdog time value is 0 ms (F_WD_Time)	Check safety PLC configuration.
E3	1 ... 4	255	28	0 ... 31	19	CRC1-Fault	Check safety PLC configuration. If this error persists, contact ABB technical support.
E3	1 ... 4	255	28	0 ... 31	28	Mismatch of safety destination address (F_Dest_Add)	Check safety PLC configuration or switch address setting. Restart safety PLC. If this error persists, replace safety PLC.
E3	1 ... 4	255	28	0 ... 31	42	Parameter "F_CRC_Length" does not match the generated values	Check safety PLC configuration.
E3	1 ... 4	255	28	0 ... 31	40	Version of F-Parameter set incorrect	Check safety PLC configuration.
E3	1 ... 4	255	30	1	17	Safety source addresses cannot be checked	Check PROFIsafe F-Host library version (2.0.0 or above). If this error persists, contact ABB technical support.
E3	1 ... 4	255	30	1	54	PROFIsafe F_Dest_Add rules are violated	Check safety PLC configuration or switch address setting against PROFIsafe F_Dest_Add configuration rules. Restart safety PLC. If this error persists, contact ABB technical support.

Table 106: Mapping of AC500/AC500-S errors to PROFINET channel errors

AC500/AC500-S error	PROFINET channel error type	PROFINET diagnostic information
28	64	Mismatch of safety destination address (F_Dest_Add)
1	65	Safety destination address not valid (F_Dest_Add)
2	66	Safety source address not valid (F_Source_Add)
11	67	Safety watchdog time value is 0 ms (F_WD_Time)
10	68	Parameter "F_SIL" exceeds SIL from specific device application
42	69	Parameter "F_CRC_Length" does not match the generated values
40	70	Version of F-Parameter set incorrect
19	71	CRC1-Fault

B.2.2 Error messages for safety I/O modules

Table 107: Error messages for safety I/O modules (channel or module reintegration is possible)

Error severity	Compo-nent or interface	Device	Module	Channel	Error	Error text	Remedy
E3	14	1..10	0	0..15	3	Discrepancy time expired	Check discrepancy time value, channel wiring and sensor.
E3	14	1..10	0	0..15	12	Test pulse error	Check wiring and sensor.
E3	14	1..10	0	0..15	13	Channel test pulse cross-talk error	Check wiring and sensor. If this error persists, replace I/O module. Contact ABB tech-nical support.
E3	14	1..10	0	0..15	25	Channel stuck-at error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
E3	14	1..10	0	0..15	28	Channel cross-talk error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
E3	14	1..10	1	0..3	4	Measurement overflow at the I/O module	Check channel wiring and sensor power supply.
E3	14	1..10	1	0..3	7	Measurement underflow at the I/O module	Check channel wiring and sensor power supply.
E3	14	1..10	1	0..3	55	Channel value difference too high	Adjust tolerance window for channels. Check channel wiring and sensor configura-tion.
E3	14	1..10	2	0..7	13	Channel read-back error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
E3	14	1..10	2	0..7	18	Channel cross-talk error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
E3	14	1..10	31	31	10	Process voltage too high	Check process voltage
E3	14	1..10	31	31	11	Process voltage too low	Check process voltage
E3	14	1..10	31	31	20	PROFIsafe communication error	Restart I/O module. If this error persists, contact ABB technical support.
E3	14	1..10	31	31	25	PROFIsafe watchdog timed out	Restart I/O module. If this error persists, increase PROFIsafe watchdog time.
E3	14	1..10	31	31	43	Internal error in the device	Replace I/O module

Table 108: Error messages for safety I/O modules (channel or module reintegration is not possible)

Error severity	Compo-nent or interface	Device	Module	Channel	Error	Error text	Remedy
E3	14	1..10	31	31	18	Plausibility check failed (iParameter)	Check configuration
E3	14	1..10	31	31	19	Checksum error in the I/O module	Check safety configuration and CRCs for I- and F-Parameters.
E3	14	1..10	31	31	26	Parameter value	Check master or configuration
E3	14	1..10	31	31	28	F-Parameter configuration and address switch value do not match.	Check I/O module F-Parameter configuration and module address switch value.

B.3 AC500 V2 non-safety CPU parameters configuration

The following parameters of non-safety CPU configuraton influence the overall system behavior of safety and non-safety CPU.

- *“Behavior of outputs in stop”*
- *“Stop on error class”*
- *“Warmstart”* after error of severity level 2

The settings for these parameters do not compromise on system safety.

“Behavior of outputs in stop”

Value *“Off in hardware and online”* (default)

If non-safety CPU is stopped, the application program execution on the safety CPU is stopped. Transferring safety CPU output values by non-safety CPU in safety telegrams will be stopped, too. No valid PROFIsafe safety telegrams can reach safety I/O modules and other F-Devices. They go to a passivation state after the watchdog time runs out.

Value *“Off in hardware and actual state online”*

If non-safety CPU is stopped, transferring safety CPU output values in PROFIsafe safety telegrams will be stopped, too. The hardware status of safety CPU communication interface becomes "0". Online display shows the last valid values from the last safety application program cycle. As a result of stopped value transfer to the safety CPU communication interface, no valid PROFIsafe safety telegrams can reach safety I/O modules and other F-Devices. They go to a passivation state after the watchdog time runs out.

Value *“Actual state in hardware and online”*

If non-safety CPU is stopped, safety CPU continues running. Safety CPU output values in PROFIsafe safety telegrams will continue to be transferred by non-safety CPU. Hardware status of the safety CPU communication interface and online display values remain intact. Safety I/O modules and other F-Devices can receive safety telegrams from the safety CPU. Operation of safety part is not influenced by the stop of non-safety CPU.

“Stop on error class”

Value *“E2”* (default)

If an error of severity level 1 or 2 occurs, non-safety CPU, all its communication modules and safety CPU will be stopped. PROFIsafe F-Host and F-Devices stacks continue running on the safety CPU with fail-safe values.

Value *“E3”*

If an error of severity level 1, 2 or 3 occurs, non-safety CPU, all its communication modules and safety CPU will be stopped. PROFIsafe F-Host and F-Devices stacks continue running on safety CPU with fail-safe values.

Value *“E4”*

If an error of severity level 1, 2, 3 or 4 occurs, non-safety CPU, all its communication modules and safety CPU will be stopped. PROFIsafe F-Host and F-Devices stacks continue running on safety CPU with fail-safe values.

“Warmstart”

Value *“Off”* (default)

If an error of severity level 2 occurs, no warm restart of non-safety CPU, all its communication modules and safety CPU will be done.

Values *“On after E2 error”*, *“On after short voltage dip”*, *“On after E2 or short voltage dip”*

If an error of severity level 2 occurs or after short voltage dip, a warm restart of non-safety CPU, all its communication modules and safety CPU will be done. After restart of safety CPU, remote safety I/O modules can be reintegrated, e.g., using PROFIsafe F-Device reintegration scheme ✎ *[3]*.

B.4 AC500 V2 non-safety CPU PLC commands

The following PLC browser commands (if supported by the current non-safety CPU firmware) from non-safety CPU can influence safety CPU state:

- `reboot`
  It reboots non-safety CPU and, as a result, safety CPU will be restarted as well.
- `resetprgorg`
  It restores non-safety and safety CPU original state (all variables, flash memory sections, etc. get original values). Safety CPU changes its state from RUN to SAFE STOP (non-safety).
- `stopprg`, `resetprg`, `resetprgcold` and menu entries *"Online ➔ Reset (cold, original)"*
  They force the safety CPU to leave RUN (safety) mode and to switch to DEBUG STOP (non-safety) mode.
- `startprg`
  It forces the safety CPU to leave DEBUG STOP (non-safety) mode and to switch to DEBUG RUN (non-safety) mode. If safety CPU is already in RUN (safety) mode or DEBUG RUN (non-safety) mode, this PLC shell command has no influence on the safety CPU.

B.5 Data exchange between safety CPU and AC500 V2 non-safety CPU

Data exchange options between safety CPU and AC500 V2 non-safety CPU:

- Acyclic non-safe data exchange: several safety CPU cycles needed to transfer the data, max. 84 bytes each direction ↬ *Appendix B.5.1 “Acyclic non-safe data exchange” on page 385*
- Cyclic non-safe data exchange: max. 3 safety CPU cycles needed to transfer the data, max. 2 kB each direction ↬ *Appendix B.5.2 “Cyclic non-safe data exchange” on page 390*

DANGER!

It is not recommended to transfer data values from non-safety CPU to safety CPU. If doing so, end-users have to define additional process-specific validation procedures in the safety program to check the correctness of the transferred non-safety data, if they would like to use those non-safety values for safety functions.

It is of no concern to transfer data values from safety CPU to non-safety CPU, e.g., for diagnosis and later visualization on operator panels.

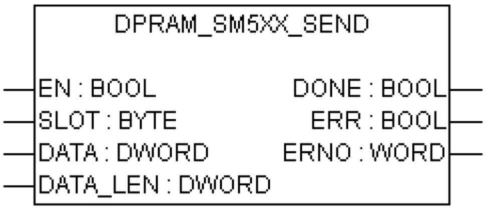
B.5.1 Acyclic non-safe data exchange

Acyclic non-safe data exchange is available per default in the programming environment, for safety CPU and non-safety CPU.

On safety CPU, use the function blocks SF_DPRAM_PM5XX_S_REC and SF_DPRAM_PM5XX_S_SEND ↳ *Chapter 4.6.7.13 "SF_DPRAM_PM5XX_S_REC" on page 320* ↳ *Chapter 4.6.7.14 "SF_DPRAM_PM5XX_S_SEND" on page 322*.

On non-safety CPU, use the function blocks DPRAM_SM5XX_SEND and DPRAM_SM5XX_REC ↳ *Appendix B.5.1.1 " DPRAM_SM5XX_SEND" on page 386* ↳ *Appendix B.5.1.2 " DPRAM_SM5XX_REC" on page 388*.

B.5.1.1 DPRAM_SM5XX_SEND



The DPRAM_SM5XX_SEND function block sends data to the safety CPU

The DPRAM_SM5XX_SEND function block is used to send data to the safety CPU. The data to be sent are available in the memory area (DATA, memory address for data to be transmitted, provided via ADR operator). The function block is activated with a TRUE signal ("0" ➔ "1" edge) at input EN. The slot number of the safety CPU is set at input SLOT. The length of the data to be transmitted is specified in bytes at input DATA_LEN. DONE = TRUE and ERR = FALSE indicate that the sending process was successful. If an error was detected during function block processing, the error is indicated at the outputs ERR and ERNO.

Note: Sending data using the DPRAM_SM5XX_SEND function block is edge-triggered, i.e. each sending process is initiated by a FALSE/TRUE edge at input EN.

Table 109: FB name: DPRAM_SM5XX_SEND

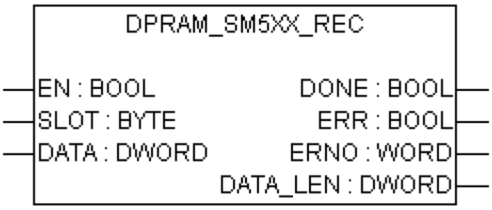
Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	Enabling of function block processing. Processing of this function block is controlled by input EN. The data transfer is initiated by a FALSE/ TRUE edge. The sending of data is indicated by output DONE.
SLOT	BYTE	16#00	Slot number (module number) Input SLOT is used to select the slot (module number) the data should be sent to. The external slots are numbered consecutively from right to left, starting with number 1.
DATA	DWORD	16#00000000	Memory address for data to be transmitted, provided via ADR operator Input DATA is used to specify the address of the variable the user data are to be copied to. The address specified at DATA has to belong to a variable of the type ARRAY or STRUCT. Note: Set the variable size to the maximum expected amount of data in order to avoid overlapping of memory areas.
DATA_LEN	WORD	16#0000	Length of data to be transmitted (in bytes) starting at address DATA, max. 84. The length of the data to be transmitted is specified in bytes at input DATA_LEN. The maximum number is 84.
VAR_OUTPUT			

Name	Data type	Initial value	Description, parameter values
DONE	BOOL	FALSE	The data was sent. Output DONE indicates that data was sent. This output always has to be considered together with output ERR. The following applies: ● DONE = TRUE and ERR = FALSE: Sending completed. A data set was sent correctly. ● DONE = TRUE and ERR = TRUE: An error occurred during sending. The error number is indicated at output ERNO.
ERR	BOOL	FALSE	Error message of the function block. Output ERR indicates whether an error occurred during sending. This output always has to be considered together with output DONE. The following applies if an error occurred during sending: DONE = TRUE and ERR = TRUE. Output ERNO indicates the error number.
ERNO	WORD	16#0000	Error number ⇘ *[4]* Output ERNO provides an error identifier if an invalid value has been applied to an input or if an error occurred during job processing. ERNO always has to be considered together with the outputs DONE and ERR. The output value at ERNO is only valid if DONE = TRUE and ERR = TRUE. The error messages encoding at output ERNO is explained at the beginning of the function block description.

Call in ST

```
SM5xxSend (EN := SM5xxSend_EN,
SLOT := SM5xxSend_SLOT,
DATA := ADR(SM5xxSend_DATA),
DATA_LEN := SM5xxSend_DATA_LEN,
DONE => SM5xxSend_DONE,
ERR => SM5xxSend_ERR,
ERNO => SM5xxSend_ERNO);
```

B.5.1.2 DPRAM_SM5XX_REC



The DPRAM_SM5XX_REC function block receives data from the safety CPU

The DPRAM_SM5XX_REC is used to receive data from the safety CPU. The data is stored in the memory area (DATA, memory address for received data, provided via ADR operator). The function block is enabled by a TRUE signal at input EN. It remains active until input EN is set to FALSE. The slot number of the safety CPU is set at input SLOT. Output DATA_LEN displays the length of the received data in bytes. DONE = TRUE and ERR = FALSE indicate that the reception was successful. If an error was detected during function block processing, the error is indicated at the outputs ERR and ERNO.

Note: Reception using the DPRAM_SM5XX_REC function block is not edge-triggered. Therefore, input EN has to be continuously set to TRUE during data reception.

Table 110: FB name: DPRAM_SM5XX_REC

Name	Data type	Initial value	Description, parameter values
VAR_INPUT			
EN	BOOL	FALSE	Enabling of function block processing. Processing of this function block is controlled by input EN. The function block is active if EN = TRUE. The reception of data is indicated by output DONE.
SLOT	BYTE	16#00	Slot number (module number) Input SLOT is used to select the slot (module number) the data should be read from. The external slots are numbered consecutively from right to left, starting with number 1.
DATA	DWORD	16#00000000	Memory address for received data, provided via ADR operator. Input DATA is used to specify the address of the variable the user data are to be copied to. The address specified at DATA has to belong to a variable of the type ARRAY or STRUCT. Note: Set the variable size to the maximum expected amount of data in order to avoid overlapping of memory areas.
VAR_OUTPUT			

Name	Data type	Initial value	Description, parameter values
DONE	BOOL	FALSE	The data was received. Output DONE indicates the reception of data. This output always has to be considered together with output ERR. The following applies: ● DONE = TRUE and ERR = FALSE: Reception completed. A data set was received correctly. ● DONE = TRUE and ERR = TRUE: An error occurred during reception. The error number is indicated at output ERNO.
ERR	BOOL	FALSE	Error message of the function block. Output ERR indicates whether an error occurred during reception. This output always has to be considered together with output DONE. The following applies if an error occurred during the processing of the function block: DONE = TRUE and ERR = TRUE. Output ERNO indicates the error number.
ERNO	WORD	16#0000	Error number *[4]* Output ERNO provides an error identifier if an invalid value was applied to an input or if an error occurred during job processing. ERNO always has to be considered together with the outputs DONE and ERR. The output value at ERNO is only valid if DONE = TRUE and ERR = TRUE. The error messages encoding at output ERNO is explained at the beginning of the function block description.
DATA_LEN	WORD	16#0000	Data length in bytes Output DATA_LEN displays the length of the received data in bytes. DATA_LEN is only valid if DONE = TRUE.

Call in ST

```
SM5xxRec (EN := SM5xxRec_EN,
SLOT := SM5xxRec_SLOT,
DATA := ADR(SM5xxRec_DATA),
DONE => SM5xxRec_DONE,
ERR => SM5xxRec_ERR,
ERNO => SM5xxRec_ERNO,
DATA_LEN => SM5xxRec_DATA_LEN);
```

B.5.2 Cyclic non-safe data exchange

In Automation Builder, use the tab *"Data exchange configuration"* of safety CPU to configure cyclic non-safe data exchange functionality. It enables data exchange between the safety CPU and non-safety CPU for a fast cyclic communication and big data amount transfer via DPRAM. In most safety applications, this functionality is not needed and shall not be used. As default, checkbox *"Cyclic non-safe data exchange"* is unselected. If you still need it, please refer to the description on how to use cyclic non-safe data exchange functionality, available via *www.abb.com/plc - document no. 3ADR025195M0202*.

Cyclic non-safe data exchange with AC500 V2 non-safety CPUs is supported from Automation Builder 1.0.1.

C Usage of safety CPU with AC500 V3 non-safety CPU PM56xx

C.1 Compatibility with AC500 V3 non-safety CPUs

All compatibility information is valid for normal and XC devices.

Table 111: Compatibility for safety CPU with AC500 V3 non-safety CPU

Safety CPU	**SM560-S**	**SM560-S-FD-1, SM560-S-FD-4**
Firmware version of safety CPU	Any	Any
Non-safety CPU	Any V3 CPU, except AC500-eCo CPUs	
Firmware version of non-safety CPU	V3.3.0 or higher	Under preparation
Version of engineering suite Automation Builder	2.3.0 or higher	

Table 112: Compatibility for AC500-S with non-safety components except CPUs

Component	**SM560-S**	**SM560-S-FD-1, SM560-S-FD-4**
Firmware version of communication module CM579-PNIO	V2.8.6.21 or higher	V2.8.6.21 or higher
Firmware version of communication module CM589-PNIO(-4)	Not applicable	Under preparation
Fimware version of communication interface module CI501-PNIO, CI502-PNIO, CI504-PNIO, CI506-PNIO	V3.2.0 or higher	V3.2.0 or higher

C.2 Error messages with AC500 V3 non-safety CPUs

C.2.1 Error messages for safety CPUs

The errors are shown as they are displayed in Automation Builder. In CODESYS Safety, errors are displayed similar to error messages of AC500 V2 non-safety CPUs.

Table 113: Error messages for safety CPU

Severity	Error code	Description	Remedy
2	8235	Internal error	Replace module
2	8448	Operation finished	Change Safety PLC switch address setting or remove SD-Card from non-safety PLC. Restart Safety PLC. If this error persists, replace Safety PLC.
2	8449	Wrong user data	Delete user data from Safety PLC. Restart Safety PLC and write user data again.
2	8450	Internal PROFIsafe initialization error	Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support.
2	8460	Flash read error	Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support.
2	8466	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8476	Boot project download error	Reload boot project. If this error persists, replace Safety PLC.
2	8488	Wrong firmware version	Update Safety PLC firmware. Restart Safety PLC. If this error persists, replace Safety PLC.
2	8491	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8496	Overvoltage or undervoltage detected	Restart Safety PLC. Check Safety PLC setting for power supply error. If this error persists, replace Safety PLC.
2	8500	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8704	User program triggered safe stop	Check user program
2	8705	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8706	Internal PROFIsafe error	Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support.
2	8707	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8714	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8717	Flash write error	Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support.

Severity	Error code	Description	Remedy
2	8721	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8722	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8723	Checksum error has occured in Safety PLC	Restart Safety PLC. If this error persists, replace Safety PLC.
2	8729	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8741	Cycle time error in Safety PLC	Check Safety PLC watchdog time.
2	8742	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8746	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8747	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8756	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8758	Internal error	Contact ABB technical support. Replace Safety PLC.
2	8990	PROFIsafe configuration error	Check F-Parameter configuration of I/O module and reload boot project
3	12561	Safety source addresses cannot be checked	Check PROFIsafe F-Host library version (2.0.0 or above). If this error persists, contact ABB technical support.
3	12570	Error in configuration data, safety PLC has not accepted configuration data, e.g., mismatch between safety and non-safety PLC configuration.	Create new configuration data for both safety and non-safety PLC again, re-create and download boot projects to both safety and non-safety PLC again.
3	12571	Error in configuration data, Safety PLC cannot read configuration data	Create boot project
3	12598	PROFIsafe F_Dest_Add rules are violated	Check Safety PLC configuration or switch address setting against PROFIsafe F_Dest_Add configuration rules. Restart Safety PLC. If this error persists, contact ABB technical support.
3	32770	Watchdog error coupler	
3	32771	Wrong firmware version of Communication Module	Update firmware
3	32772	Initialisation of Safety Module on slot failed. More than one Safety Module plugged	Remove this module or Only that one Safety Module plugged -> defective, replace this module
3	32774	Invalid configuration data	Check configuration

Severity	Error code	Description	Remedy
3	32775	Safety Module not found	Check configuration. At Safety PLC: Check Safety PLC switch address setting. Restart Safety PLC. If this error persists, replace Safety PLC.
3	32776	Safety Module has wrong type	Check configuration
4	16640	Reserved switch address setting.	Warning
4	16644	Boot project not loaded, maximum power dip reached	Restart Safety PLC
4	16648	Power dip data missed or corrupted. Default power dip data was flashed by Safety PLC	Warning
4	16659	CRC error boot project	Create new boot project and restart Safety PLC
4	16909	Flash write error (production data)	Warning
4	16935	More than one instance of SF_WDOG_TIME_SET or SF_MAX_POWER_DIP_SET	Warning
4	16922	No or wrong configuration data from PM5x, run state not possible	Create correct boot project at PM5x
4	17421	Flash write error (boot project)	Warning
4	17677	Flash write error (boot code)	Warning
4	17933	Flash write error (firmware)	Warning
4	18189	Flash write error (password)	Warning
4	18445	Flash write error (user data)	Warning
4	18701	Flash write error (user data)	Warning
4	18957	Flash write error (internal)	Warning
4	19213	Flash write error (internal)	Warning
4	19469	Flash write error (internal)	Warning
4	32777	Program not started because of configuration error	Check configuration
4	32778	Program not started, no application running in Safety Module	Check configuration, download safety application to Safety Module

C.2.2 Error messages for safety I/O modules

Table 114: Error messages for safety I/O modules (channel or module reintegration is possible)

Severity	Error code	Description	Remedy
3	3	Discrepancy time expired	Check discrepancy time value, channel wiring and sensor.
3	12	Test pulse error	Check wiring and sensor.
3	13	Channel test pulse cross-talk error	Check wiring and sensor. If this error persists, replace I/O module. Contact ABB technical support.
3	25	Channel stuck-at error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
3	28	Channel cross-talk error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
3	260	Measurement overflow at the I/O module	Check channel wiring and sensor power supply.
3	263	Measurement underflow at the I/O module	Check channel wiring and sensor power supply.
3	311	Channel value difference too high	Adjust tolerance window for channels. Check channel wiring and sensor configuration.
3	525	Channel readback error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
3	530	Channel cross-talk error	Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.
3	16138	Process voltage too high	Check process voltage
3	16139	Process voltage too low	Check process voltage
3	16148	PROFIsafe communica-tion error	Restart I/O module. If this error persists, contact ABB technical support.
3	16153	PROFIsafe watchdog timed out.	Restart I/O module. If this error persists, increase PROFIsafe watchdog time.
3	16171	Internal error in the device	Replace I/O module

Table 115: Error messages for safety I/O modules (channel or module reintegration ist not possible)

Severity	Error code	Description	Remedy
3	16146	Plausibility check failed (iParameter)	Check configuration
3	16147	Checksum error in the I/O module	Check safety configuration and CRCs for I- and F-Parameters.

Severity	Error code	Description	Remedy
3	16154	Parameter value	Check master or configuration
3	16156	F-Parameter configuration and address switch value do not match.	Check I/O module F-Parameter configuration and module address switch value.

C.3 AC500 V3 non-safety CPU parameters configuration

If non-safety CPU is stopped, the safety CPU will go to DEBUG STOP (non-safety) state (Fig. 12 on page 45) and safety I/O modules will immediately switch to RUN (module passivation with a command) state (Fig. 15 on page 53).

Later, if the safety CPU changes to DEBUG RUN (non-safety) state, e.g., after switching non-safety CPU back to RUN state, the safety I/O modules will immediately change to RUN (ok) state (Fig. 15 on page 53) and deliver valid process values to the safety CPU without the need for reintegration.

> **!**
>
> **NOTICE!**
>
> The described behavior with AC500 V3 non-safety CPUs is different to the behavior with AC500 V2 non-safety CPUs. If you are familiar with AC500 V2 non-safety CPUs, you need to know the following differences:
>
> If AC500 V2 non-safety CPU is stopped, the safety CPU will go to DEBUG STOP (non-safety) state and **safety I/O modules will go to RUN (module passivation) state** (Fig. 15 on page 53).
>
> If the safety CPU changes to DEBUG RUN (non-safety) state, the **safety I/Os have to be reintegrated first** by going through the RUN (user acknowledgement request) state (Fig. 15 on page 53) and only then deliver current valid process outputs to the safety CPU.

The following settings of AC500 non-safety module configuraton influence the overall system behavior of safety and non-safety CPUs.

Settings for non-safety CPU in Automation Builder:

- Tab *“PLC Settings”*
  - *“Bus cycle task”*
- Tab *“CPU-Parameters Parameters”*
  - *“Stop on error class”*
- Tab *“I/O-Bus I/O Mapping”*
  - *“Bus cycle task”*

Settings for communication module in Automation Builder:

- Tab *“PROFINET-IO-Controller I/O Mapping”* / *“PROFINET-IO-Device I/O Mapping”*
  - *“Bus cycle task”*

The settings for these parameters do not compromise on system safety.

"Bus cycle task"

We strongly recommend to read the AC500 user documentation ↳ *[4]* on this topic to get an understanding of parameter *"Bus cycle task"* for the above listed settings and dependencies on other parameters.

The settings have to be considered carefully. On the one hand, to avoid any overload scenarios on the non-safety CPU. On the other hand, not to exceed the SFRT.

One easy possibility to set up the bus cycle

1. Set a global bus cycle time in tab *"PLC Settings"* by assigning *"Bus cycle task"* with a task.
2. Keep the default values for the bus cycle task for I/O bus and communication modules.

With these settings, both bus cycle times for I/O bus and communication modules are driven from the non-safety CPU with the cycle time of the assigned task (in tab *"PLC Settings"*).

> **NOTICE!**
> The value of safety CPU parameter *"Update cycle time"* is the limitating bus cycle time for I/O bus and communication modules. If higher values for the bus cycle tasks are assigned for I/O bus and communication module, they will be limited to the lower value of *"Update cycle time"*. If lower values for the bus cycle tasks are assigned for I/O bus and communication module, they will be kept as they are.

> **NOTICE!**
> The cycle times for I/O bus and communication modules affect the SFRT of your system ↳ *Chapter 5.3 "Safety function response time" on page 325*.

"Stop on error class"

Parameter in tab *"CPU-Parameters Parameters"* of non-safety CPU.

Value *"Diagnosis of at least error class 2"* (default)

If an error of severity level 1 or 2 occurs, non-safety CPU and safety CPU will be stopped. If present on the given safety CPU, PROFIsafe F-Host and F-Device stacks continue running on the safety CPU with fail-safe values.

Value *"Diagnosis of at least error class 3"*

If an error of severity level 1, 2 or 3 occurs, non-safety CPU and safety CPU will be stopped. If present on the given safety CPU, PROFIsafe F-Host and F-Device stacks continue running on safety CPU with fail-safe values.

Value *"Diagnosis of at least error class 4"*

If an error of severity level 1, 2, 3 or 4 occurs, non-safety CPU and safety CPU will be stopped. If present on the given safety CPU, PROFIsafe F-Host and F-Device stacks continue running on safety CPU with fail-safe values.

C.4 AC500 V3 non-safety CPU PLC commands

The following PLC shell commands (if supported by the current non-safety CPU firmware) from non-safety CPU can influence safety CPU state:

- `reboot`
  It reboots non-safety CPU and, as a result, safety CPU will be restarted as well.
- `stopprg`, `resetprg`, `resetprgcold`
  They force the safety CPU to leave RUN (safety) mode and to switch to DEBUG STOP (non-safety) mode.
- `startprg`
  It forces the safety CPU to leave DEBUG STOP (non-safety) mode and to switch to DEBUG RUN (non-safety) mode. If safety CPU is already in RUN (safety) mode or DEBUG RUN (non-safety) mode, this PLC shell command has no influence on the safety CPU.

> **NOTICE!**
> The error messages of the safety CPU are aggregated in the diagnosis system on non-safety CPU. For handling and usage of the diagnosis features of the non-safety CPU, refer to ☞ *[4]*.

C.5 Data exchange between safety CPU and AC500 V3 non-safety CPU

Data exchange options between safety CPU and AC500 V3 non-safety CPU:

- Acyclic non-safe data exchange: several safety CPU cycles needed to transfer the data, max. 84 bytes each direction ↬ *Appendix C.5.1 “Acyclic non-safe data exchange” on page 401*
- Cyclic non-safe data exchange: max. 3 safety CPU cycles needed to transfer the data, max. 2 kB each direction ↬ *Appendix C.5.2 “Cyclic non-safe data exchange” on page 402*

DANGER!

It is not recommended to transfer data values from non-safety CPU to safety CPU. If doing so, end-users have to define additional process-specific validation procedures in the safety program to check the correctness of the transferred non-safety data, if they would like to use those non-safety values for safety functions.

It is of no concern to transfer data values from safety CPU to non-safety CPU, e.g., for diagnosis and later visualization on operator panels.

C.5.1 Acyclic non-safe data exchange

On safety CPU, use the function blocks SF_DPRAM_PM5XX_S_REC and SF_DPRAM_PM5XX_S_SEND ↪ *Chapter 4.6.7.13 "SF_DPRAM_PM5XX_S_REC" on page 320*. ↪ *Chapter 4.6.7.14 "SF_DPRAM_PM5XX_S_SEND" on page 322*

On non-safety CPU, use the function blocks Sm560Send and Sm560Rec. The function blocks are included in library SM560Safety. In Automation Builder, refer to Library Manager to get a detailed description.

C.5.2 Cyclic non-safe data exchange

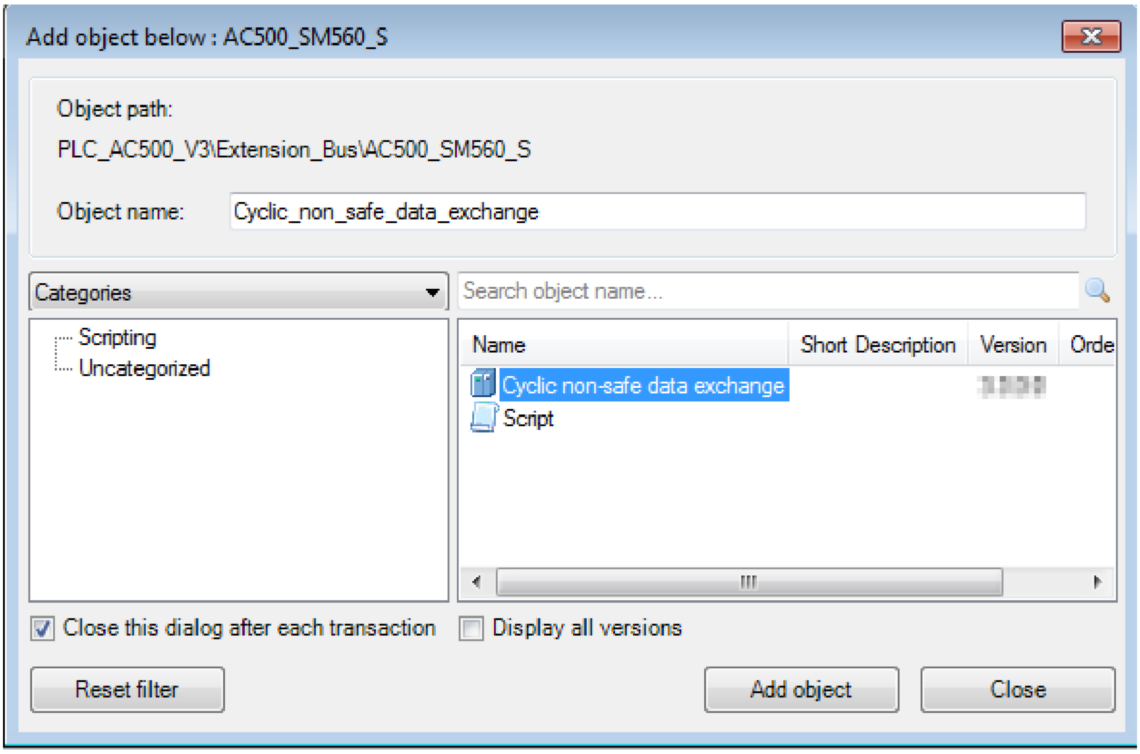
DANGER!
If cyclic non-safe data exchange is used to receive or send safety data from or to safety CPU, then SIL 3 (IEC 61508 and IEC 62061) and PL e (ISO 13849-1) functional safety requirements will not be fulfilled for received and sent data (independently on application safety communication profile used), because only one microprocessor (no 1oo2 safety architecture in the background) on safety CPU handles the sending and receiving direction.

Contact ABB technical support on how to reach SIL 3 and PL e.

DANGER!
It must be guaranteed by proper Automation Builder user management configuration that only users of the safety group are allowed to implement cyclic non-safe data exchange.

How to use cyclic non-safe data exchange

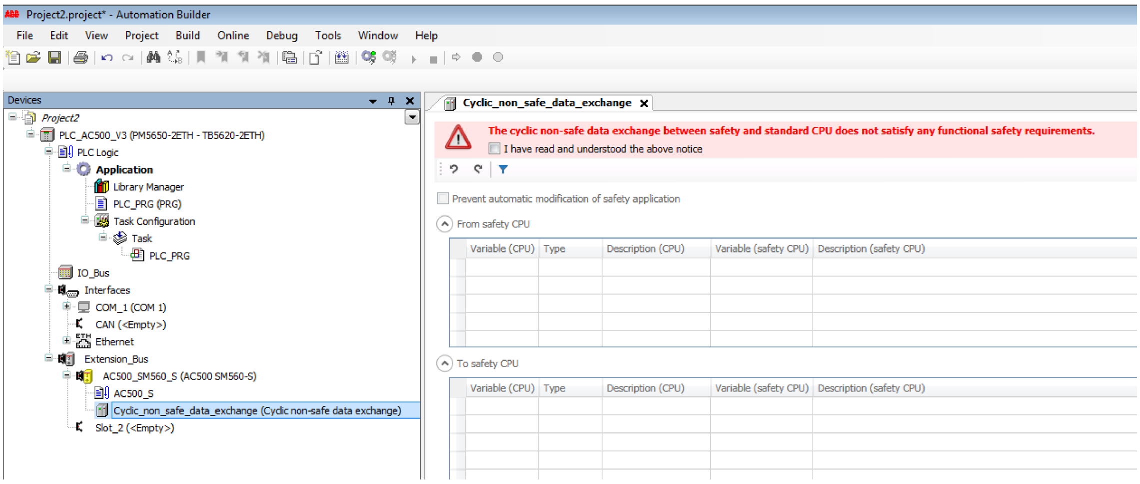
1. Right-click on the safety CPU node and select *“Add object”*.
2. Select *“Cyclic non-safe data exchange”*.



⇨ Cyclic non-safe data exchange instance is added to the safety CPU node.

3. Double-click on the *“Cyclic non-safe data exchange”* instance.

   ⇨ A warning is displayed that safety requirements are not fulfilled when using the cyclic non-safe data exchange.



4. Carefully read the warning and confirm it.

   Without confirming, you are not able to define variables and therefore not able to use the data exchange.

5. For details on checkbox *“Prevent automatic modification of safety application”* refer to *Appendix C.5.2.1 “Migration from AC500 V2 to AC500 V3 (compatibility mode)” on page 407*.

6. Define variables in the tables. Refer to the detailed description for defining variables *“Define variables” on page 404*.

   Table *“From safety CPU”*: Variables which shall be written by the safety CPU and read by the non-safety CPU.

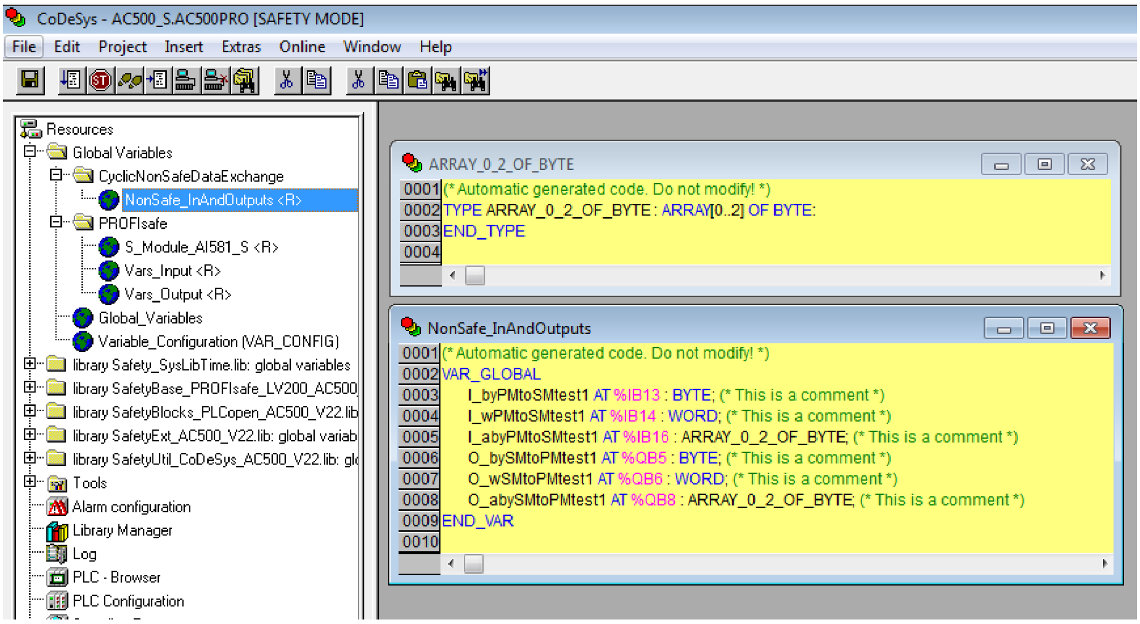
   Table *“To safety CPU”*: Variables which shall be written by the non-safety CPU and read by the safety CPU.

7. Build or rebuild the non-safety application in Automation Builder. Do this after each modification for cyclic non-safe data exchange, e.g., new variables added or existing variables updated.

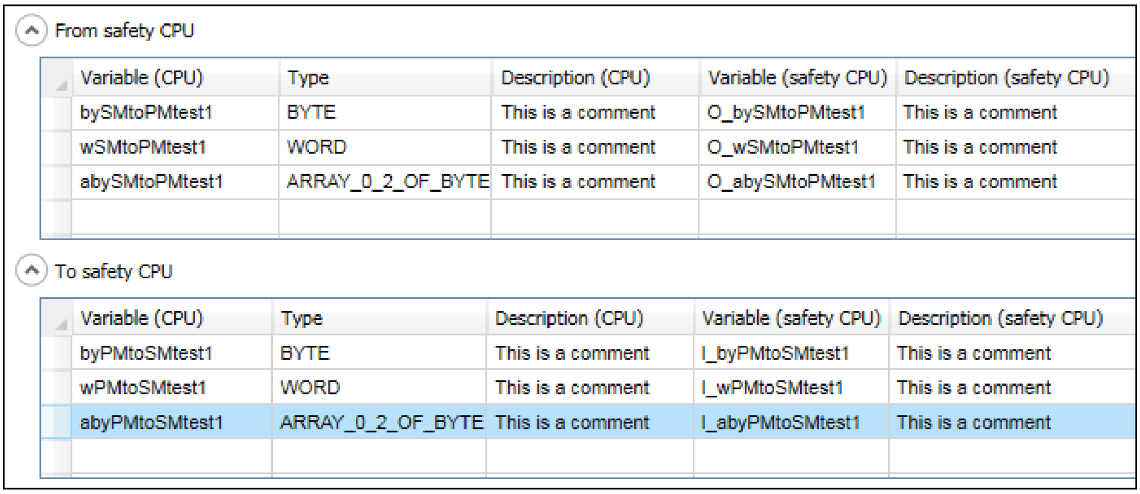
   ⇨ The variables are created and can be used in non-safety application.

8. Right-click on the safety application node (*“AC500_S”*) and select *“Create Safety Configuration Data”*. Do this after each modification for cyclic non-safe data exchange, e.g., new variables added or existing variables updated.

   ⇨ The variables are created and can be used in CODESYS Safety.



Define variables



From safety CPU

Variable (CPU)	Type	Description (CPU)	Variable (safety CPU)	Description (safety CPU)
bySMtoPMtest1	BYTE	This is a comment	O_bySMtoPMtest1	This is a comment
wSMtoPMtest1	WORD	This is a comment	O_wSMtoPMtest1	This is a comment
abySMtoPMtest1	ARRAY_0_2_OF_BYTE	This is a comment	O_abySMtoPMtest1	This is a comment

To safety CPU

Variable (CPU)	Type	Description (CPU)	Variable (safety CPU)	Description (safety CPU)
byPMtoSMtest1	BYTE	This is a comment	I_byPMtoSMtest1	This is a comment
wPMtoSMtest1	WORD	This is a comment	I_wPMtoSMtest1	This is a comment
abyPMtoSMtest1	ARRAY_0_2_OF_BYTE	This is a comment	I_abyPMtoSMtest1	This is a comment

Variable (CPU)	Variable name for non-safety application
Type	Variable type both for non-safety and CODESYS Safety application
Description (CPU)	Variable description for non-safety application
Variable (safety CPU)	Variable name for CODESYS Safety application
Description (safety CPU)	Variable description for CODESYS Safety application

▷ Add a variable for non-safety application in the last empty row.

   ⇨ The corresponding variable name and description for safety CPU will be added automatically. If required, you can adapt them independently from the non-safety variable name and description.

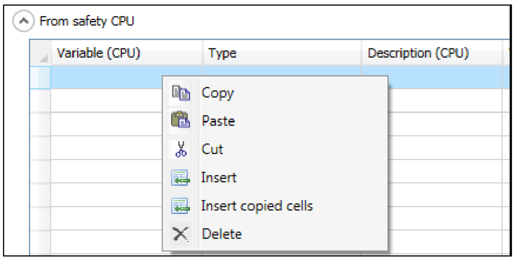
   To synchronize them again, manually change those entries which shall be the same so that variable names are written in the same way. The automatic synchronization is active again.

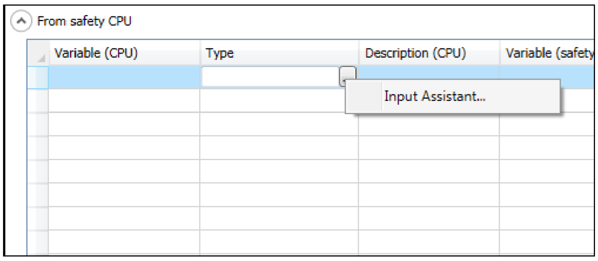
Supported data types:

- Standard data types like BYTE, WORD, INT
- Array data types
- Data unit types (DUTs)
  DUT objects are automatically created in CODESYS Safety during *"Create Safety Configuration Data"*.
- A mixture of the above

Supported features for adding variables:

- Cut, copy, paste, delete and insert of variables via context menu and standard windows shortcuts.
- Bulk data modification, e.g., copy and paste variables from and to *.csv* file.
- Filters for each column.
- Undo and redo of changes.
- *"Input Assistant"* for variable name and type *[4]*.





NOTICE!
Since the variable names are generated for both safety and non-safety application, it is recommended to use variable names that clearly describe the transmission direction, e.g., "PMtoSM" and "SMtoPM" or "toSM" and "fromSM".

DANGER!
To satisfy the CODESYS Safety programming guidelines *Chapter 4.4 "CODESYS Safety programming guidelines" on page 179*, you must follow these rules:

– Use the prefixes "I_" (non-safety inputs for the safety CPU) and "O_" (non-safety outputs from the safety CPU) for the variable names of the safety CPU. The cyclic non-safe data exchange is non-safe. Therefore, do not use any safety prefixes *Chapter 4.5 "Safety code analysis tool" on page 188*.
– Add a description for each variable with at least 10 characters.

NOTICE!
If you use cyclic non-safe data exchange, changes in non-safety programming environment could lead to new boot project CRC.

NOTICE!
Cyclic non-safe data exchange shares the memory with the PROFIsafe process data (e.g., safety inputs and outputs) of the configured safety I/O devices, and is limited to 2048 bytes for each direction.

Automation Builder does not check the size when defining the variables, but during *"Create Safety Configuration Data"*.

> **NOTICE!**
> Using cyclic non-safe data exchange influences the cycle time of non-safety CPU. E.g., data exchange with granular variables can generate a significant load on non-safety CPU.

C.5.2.1 Migration from AC500 V2 to AC500 V3 (compatibility mode)

You can migrate an existing Automation Builder project with AC500 V2 non-safety CPUs and safety CPUs with cyclic non-safe data exchange to a project with AC500 V3 non-safety CPUs. If you do not want to change the safety application, enable the checkbox *"Prevent automatic modification of safety application"*. When the checkbox is enabled, no variable assignments between safety and non-safety CPU are done.

In CODESYS Safety, no folder *"CyclicNonSafeDataExchange"* and no corresponding global variables are generated. The safety application remains unchanged. On safety CPU, data exchange with non-safety CPU is done with specific function blocks. Refer to the corresponding description, available via *www.abb.com/plc - document no. 3ADR025195M0202*.

On non-safety CPU, data exchange with safety CPU is done via the variables defined in tables *"From safety CPU"* and *"To safety CPU"*.

NOTICE!

If you use the compatibility mode *Appendix C.5.2.1 "Migration from AC500 V2 to AC500 V3 (compatibility mode)" on page 407*, use the checklist for cyclic non-safe data exchange with AC500 V2 *Appendix B.5.2 "Cyclic non-safe data exchange" on page 390*.

C.5.2.2 Troubleshooting

NOTICE!
If you use the compatibility mode *Appendix C.5.2.1 “Migration from AC500 V2 to AC500 V3 (compatibility mode)” on page 407*, refer also to the troubleshooting for cyclic non-safe data exchange with AC500 V2 *Appendix B.5.2 “Cyclic non-safe data exchange” on page 390*.

ID	Behavior	Potential cause	Remedy
1.	Cyclic non-safe variables not updated in safety and/or non-safety CPU.	Configuration has not been updated.	Clean and build/rebuild non-safety CPU application. Create safety configuration data. Check for error messages. Login to non-safety and safety CPU and download the applications. Create new boot projects for safety CPU and non-safety CPU.
2.	Safety CPU cycle time too high for the given application.	Amount of cyclic non-safe data is too big.	Check if configured variables are really necessary for the particular use case. Reduce the number of variables to increase the performance.
3.	Variable can’t be used in application because it is not defined or variable is not listed in "Input Assistant".	Configuration has not been updated.	Clean and build/rebuild non-safety CPU application. Create safety configuration data. Check for error messages.
4.	The used size of a variable is bigger than expected.	In some cases, one or more padding bytes are required to fulfill the data alignment. This is done automatically in Automation Builder.	Reorganize the variables in the used DUTs. Try to use the biggest data type at first. Bad example: ● VAR0 : BYTE ● VAR1 : DWORD ● VAR2 : BYTE ● VAR3 : WORD Good example: ● VAR1 : DWORD ● VAR3 : WORD ● VAR0 : BYTE ● VAR2 : BYTE
5.	Build errors.	Inconsistent internal data.	Clean the non-safety application and build it again.
6.	Variable in the table is not added.	Missing or wrong values for the variable definition.	Enter at least variable name *“Variable (CPU)”* and type. These values are mandatory.

ID	Behavior	Potential cause	Remedy
7.	Error message "...no valid assignment target"	Variable is defined in the wrong table.	Take care that the variables defined in the table *“From safety CPU”* are written by the safety CPU and can only be read by the non-safety CPU. Variables defined in the table *“To safety CPU”* are written by the non-safety CPU and can only be read by the safety CPU.
8.	Error message about memory overflow.	Cyclic non-safe data exchange shares the memory with the PROFIsafe process data (e.g., safety inputs and outputs) of the configured safety I/O modules and is limited to 2048 bytes in total for each direction. The Automation Builder does not check the size when defining the variables, but during *“Create Safety Configuration Data”*.	Reduce the size for cyclic non-safe data exchange and perform *“Create Safety Configuration Data”* again.

If a problem persists, contact ABB technical support.

—
ABB Automation Products GmbH
Eppelheimer Str. 82
69123 Heidelberg, Germany
Telephone: +49 (0)6221 701 1444
Fax: +49 (0)6221 701 1382
E-mail: plc.support@de.abb.com
abb.com/plc

3ADR025091M0209, 13, en_US